Exhibit D

US005299313A

United States Patent [19]

Petersen et al.

[11] Patent Number: 5,299,313

[45] Date of Patent: Mar. 29, 1994

[54] **NETWORK INTERFACE WITH HOST INDEPENDENT BUFFER MANAGEMENT**

[75] Inventors: **Brian Petersen**, Los Altos; **W. Paul Sherer**, Sunnyvale; **David R. Brown**, San Jose; **Lai-Chin Lo**, Campbell, all of Calif.

[73] Assignee: **3COM Corporation**, Santa Clara, Calif.

[21] Appl. No.: **921,519**

[22] Filed: **Jul. 28, 1992**

[51] **Int. Cl.**[5] **G06F 13/00**

[52] **U.S. Cl.** **395/200**; 395/250; 395/275

[58] **Field of Search** ..... 364/DIG. 1, DIG. 2; 395/200, 275, 250

[56] **References Cited**

U.S. PATENT DOCUMENTS

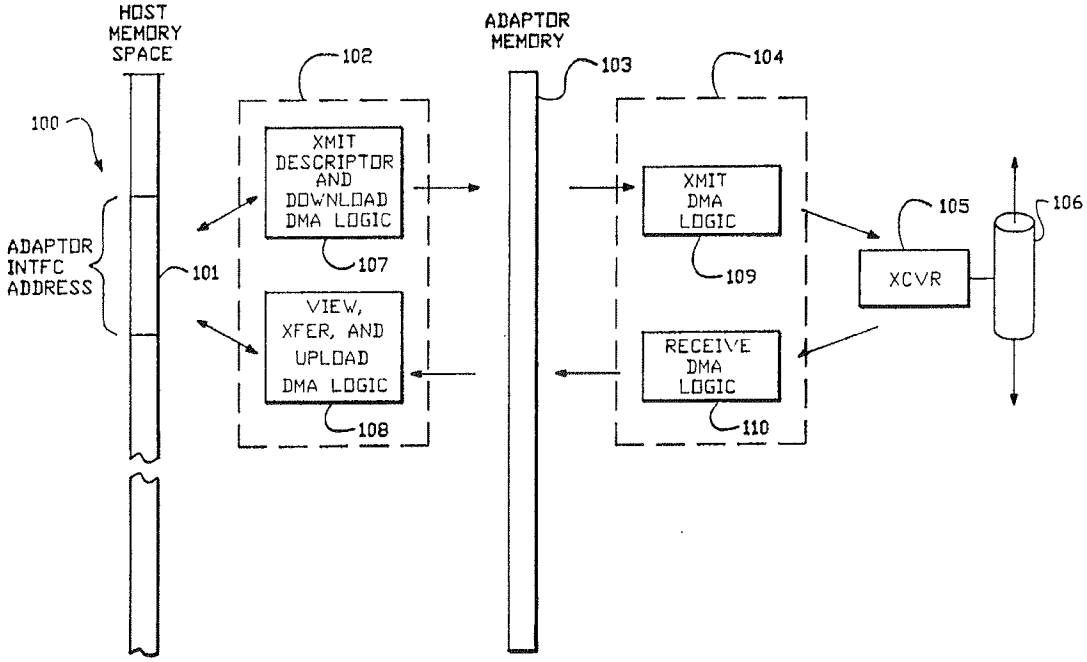
4,672,570 6/1987 Beaken 395/200

Primary Examiner—Robert L. Richardson
Attorney, Agent, or Firm—Mark A. Haynes

[57] **ABSTRACT**

A network interface controller controls communication between a host system and a network transceiver coupled to a network comprises a memory outside of the host address space in which receive and transmit buffers are managed, host interface logic emulating memory mapped registers in the host address space, for transferring data between the host address space and the buffer memory, and network interface logic coupled with the network transceiver, for transferring data between the buffers in the buffer memory and the network transceiver. The buffer memory includes a transmit descriptor ring buffer, transmit data buffer, transfer descriptor buffer, and receive ring buffer all managed by operations transparent to the host.

29 Claims, 10 Drawing Sheets



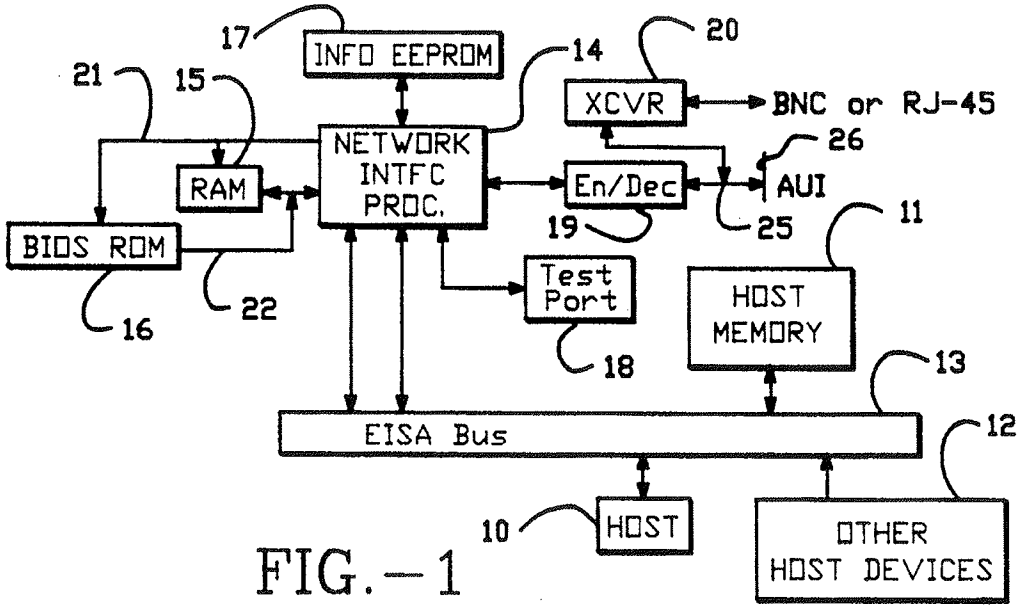

17
INFO EEPROM
14
20
21
15
XCVR
BNC or RJ-45
NETWORK INTFC PROC.
26
RAM
En/Dec
AUI
19
25
11
BIOS ROM
22
Test Port
HOST MEMORY
16
18
13
EISA Bus
12
10
HOST
OTHER HOST DEVICES

FIG.—1

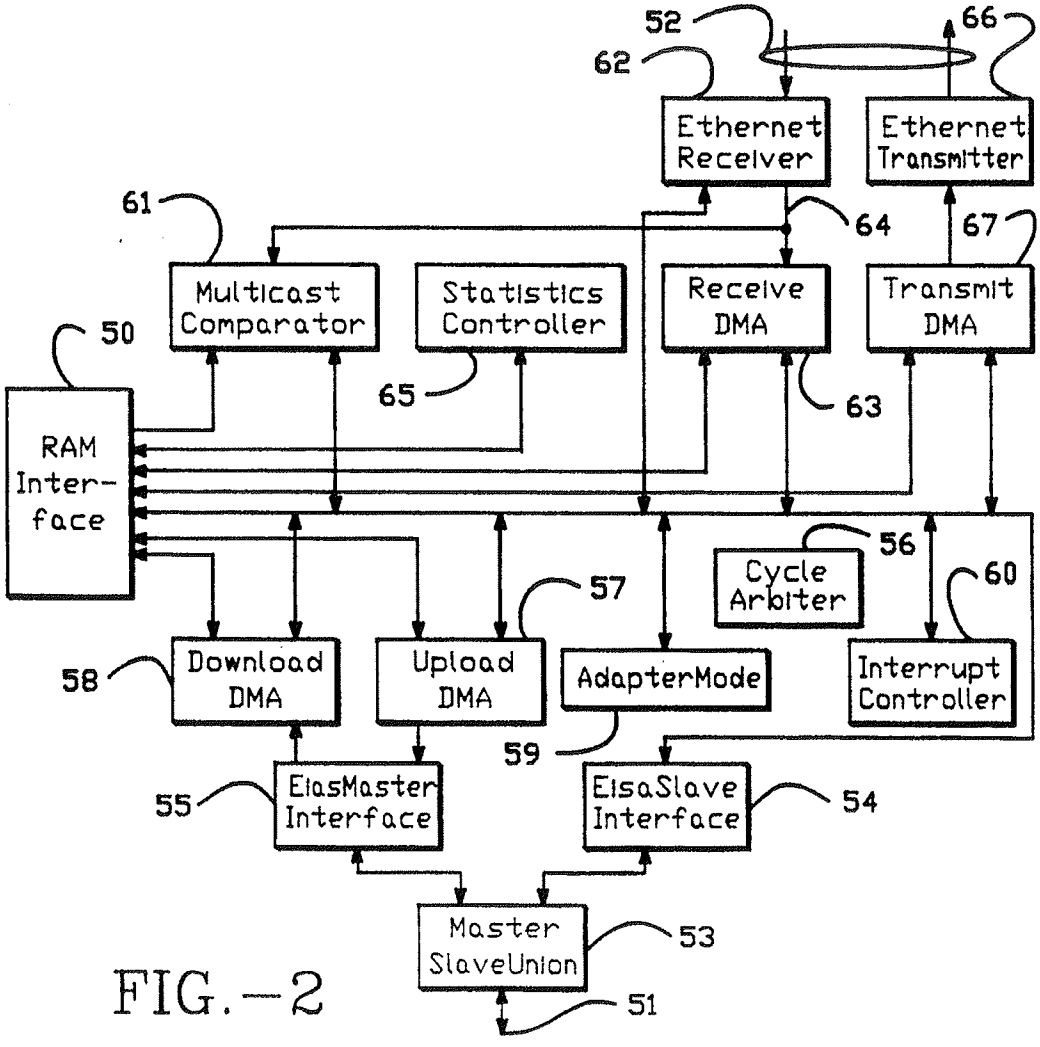

52
66
62
Ethernet Receiver
Ethernet Transmitter
61
64
67
Multicast Comparator
Statistics Controller
Receive DMA
Transmit DMA
50
65
63
RAM Inter-face
56
Cycle Arbiter
60
57
58
Download DMA
Upload DMA
AdapterMode
Interrupt Controller
59
55
EisaMaster Interface
EisaSlave Interface
54
Master SlaveUnion
53
51

FIG.—2

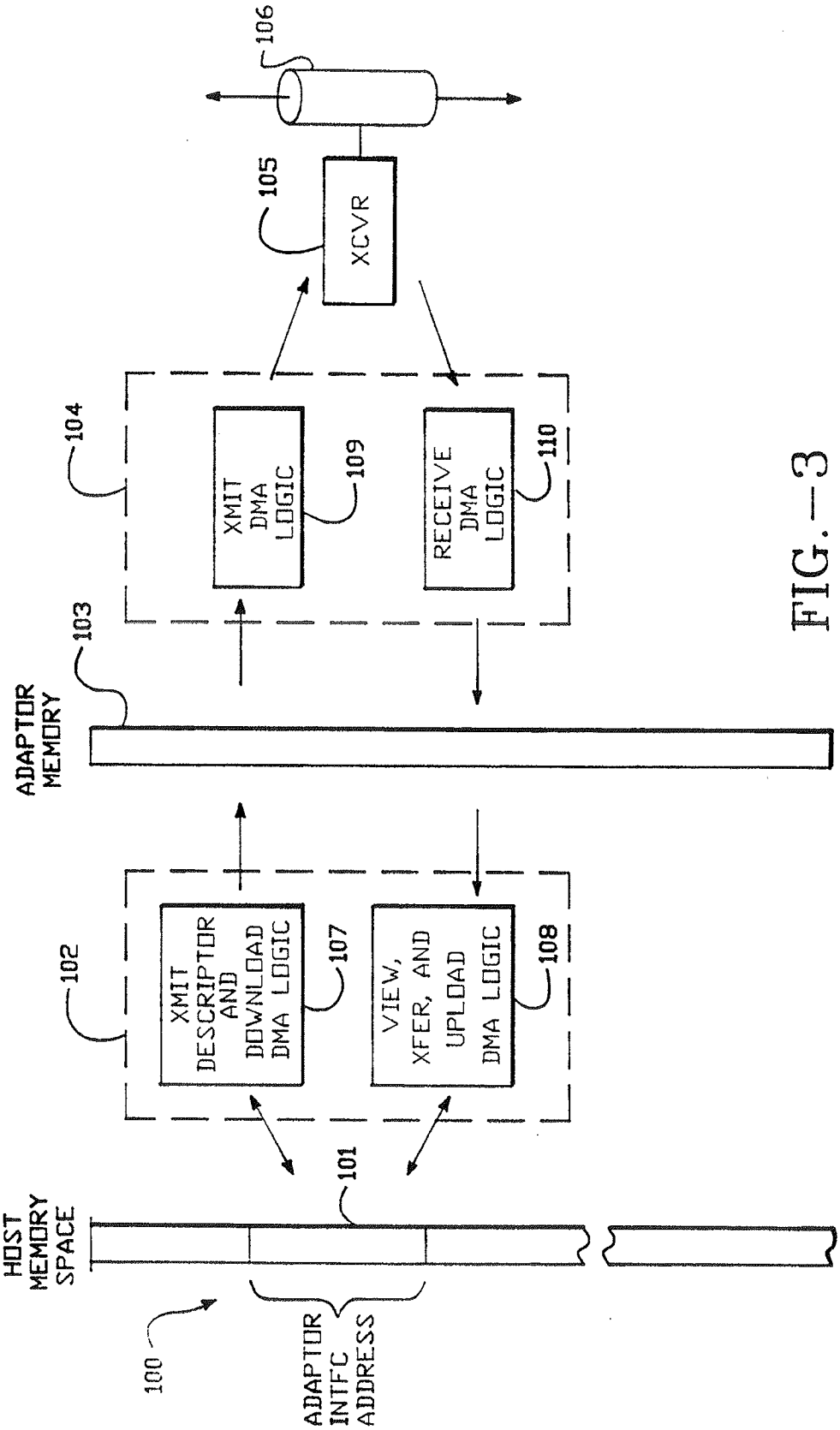

106
105
XCVR
104
XMIT DMA LOGIC
109
RECEIVE DMA LOGIC
110
103
ADAPTOR MEMORY
102
XMIT DESCRIPTOR AND DOWNLOAD DMA LOGIC
107
VIEW, XFER, AND UPLOAD DMA LOGIC
108
101
HOST MEMORY SPACE
100
ADAPTOR INTFC ADDRESS

FIG.—3

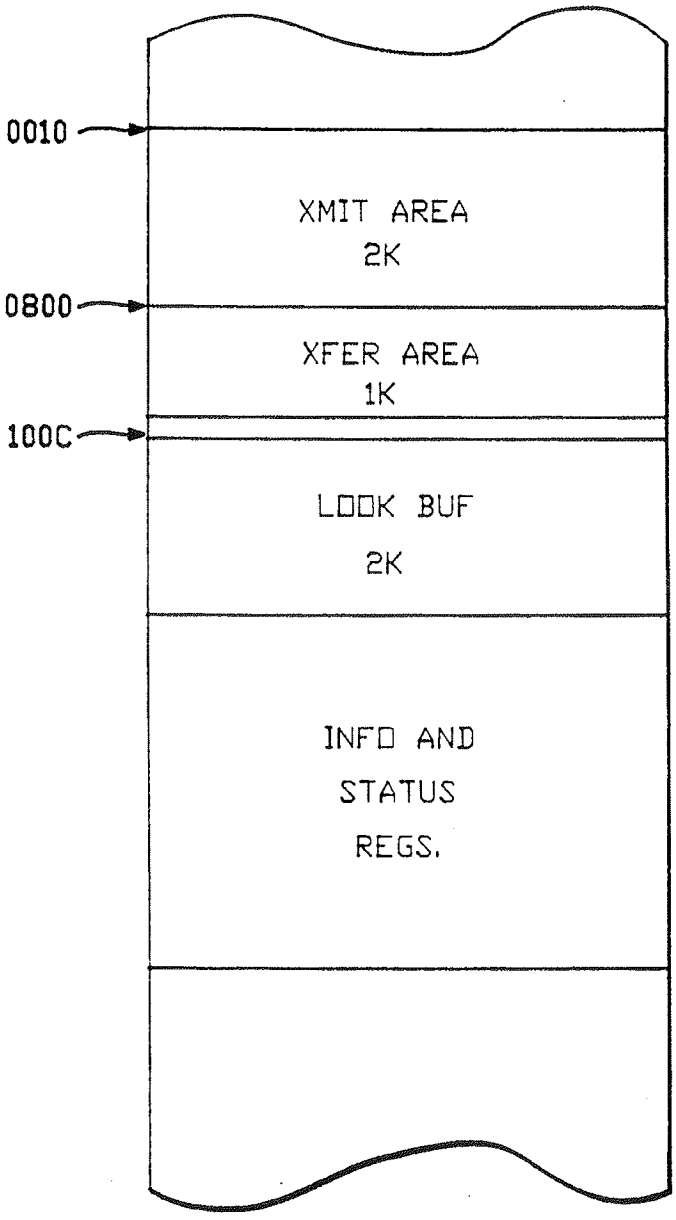
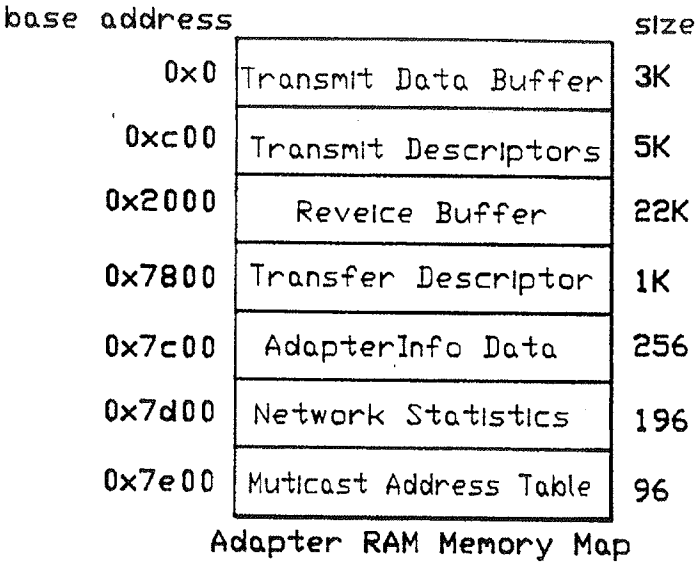

0010
XMIT AREA
2K
0800
XFER AREA
1K
100C
LOOK BUF
2K
INFO AND
STATUS
REGS.

FIG.—4



Adapter RAM Memory Map

FIG.—5

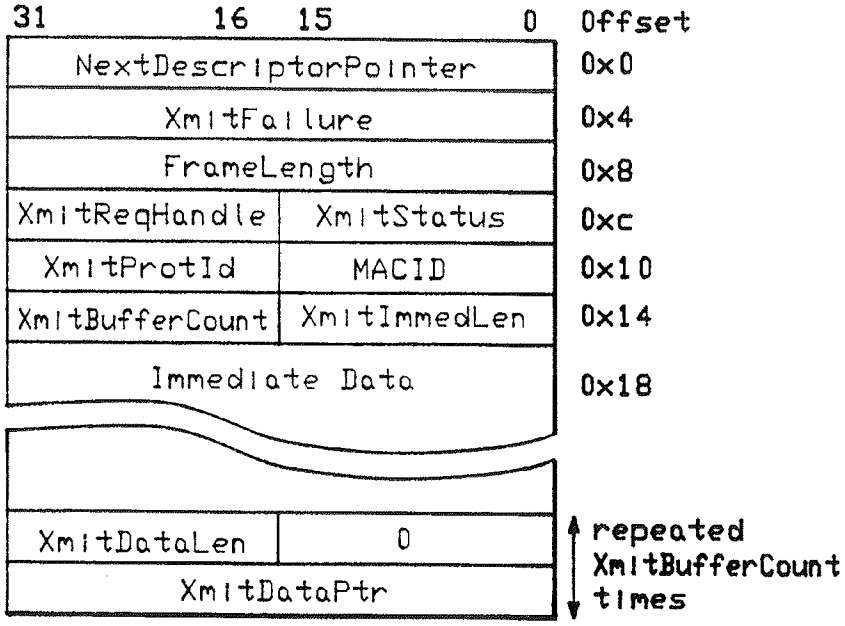


FIG.—6

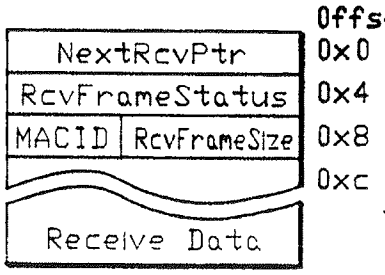


FIG.—7

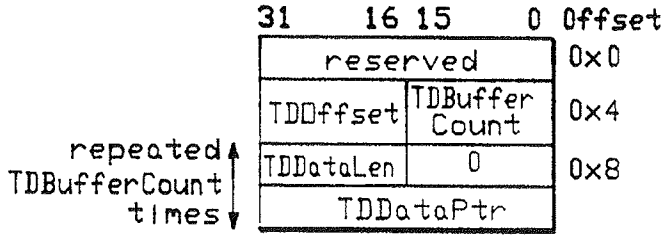


FIG.—8

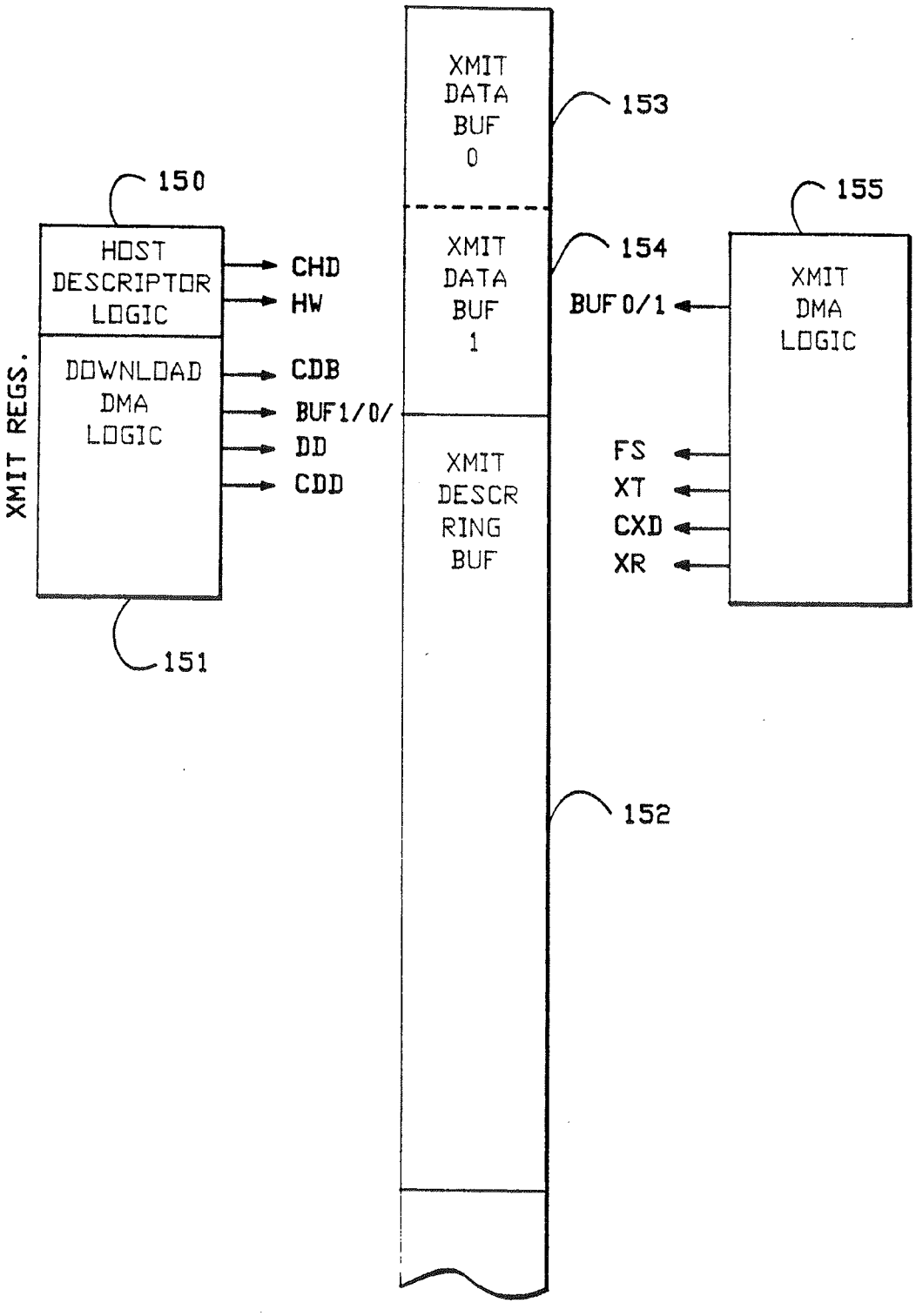

XMIT DATA BUF 0
153
150
155
HOST DESCRIPTOR LOGIC
CHD
HW
XMIT DATA BUF 1
154
BUF 0/1
XMIT DMA LOGIC
DOWNLOAD DMA LOGIC
CDB
BUF1/0/
DD
CDD
XMIT REGS.
XMIT DESCR RING BUF
FS
XT
CXD
XR
151
152

FIG.—9

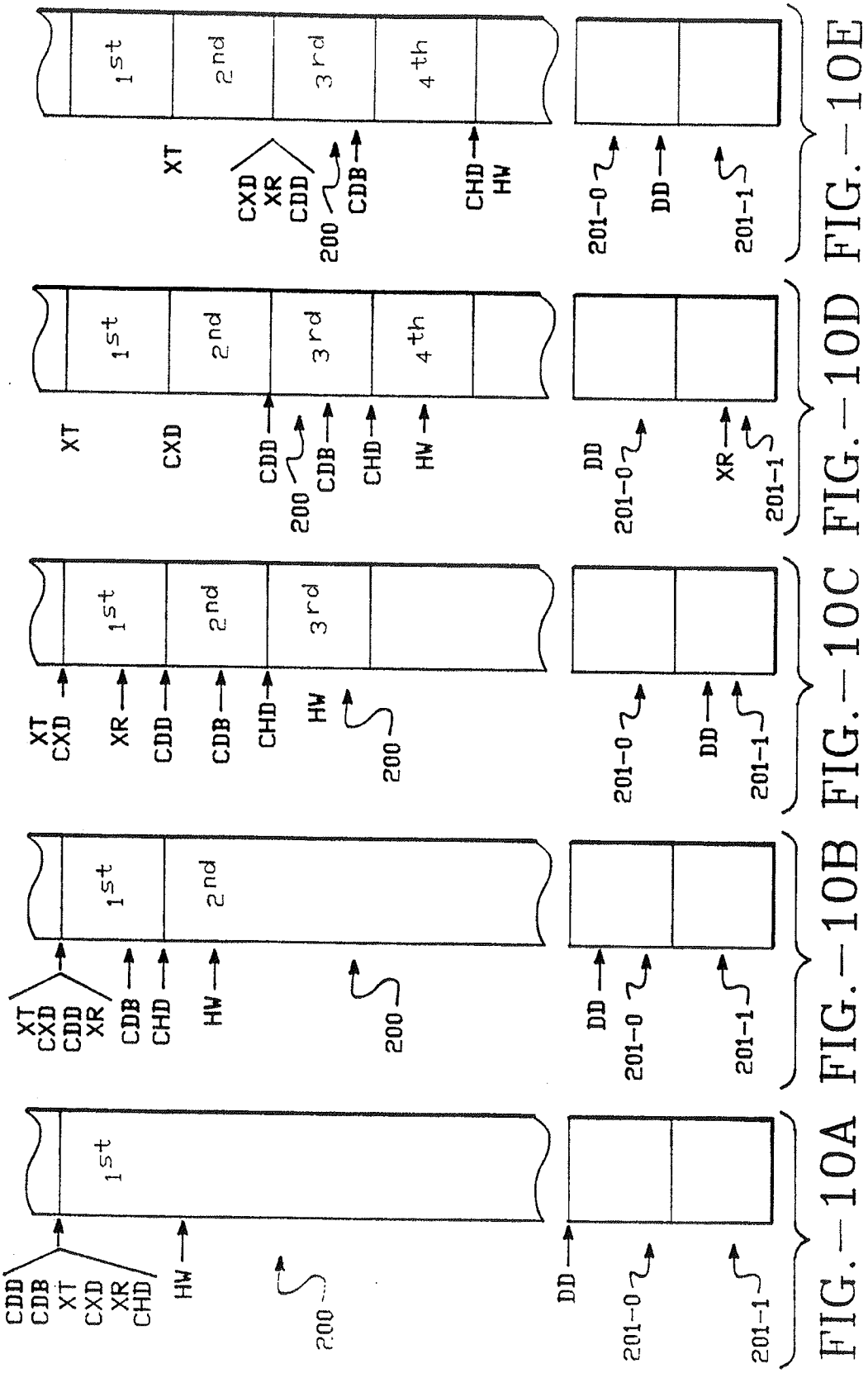


FIG.—10A FIG.—10B FIG.—10C FIG.—10D FIG.—10E

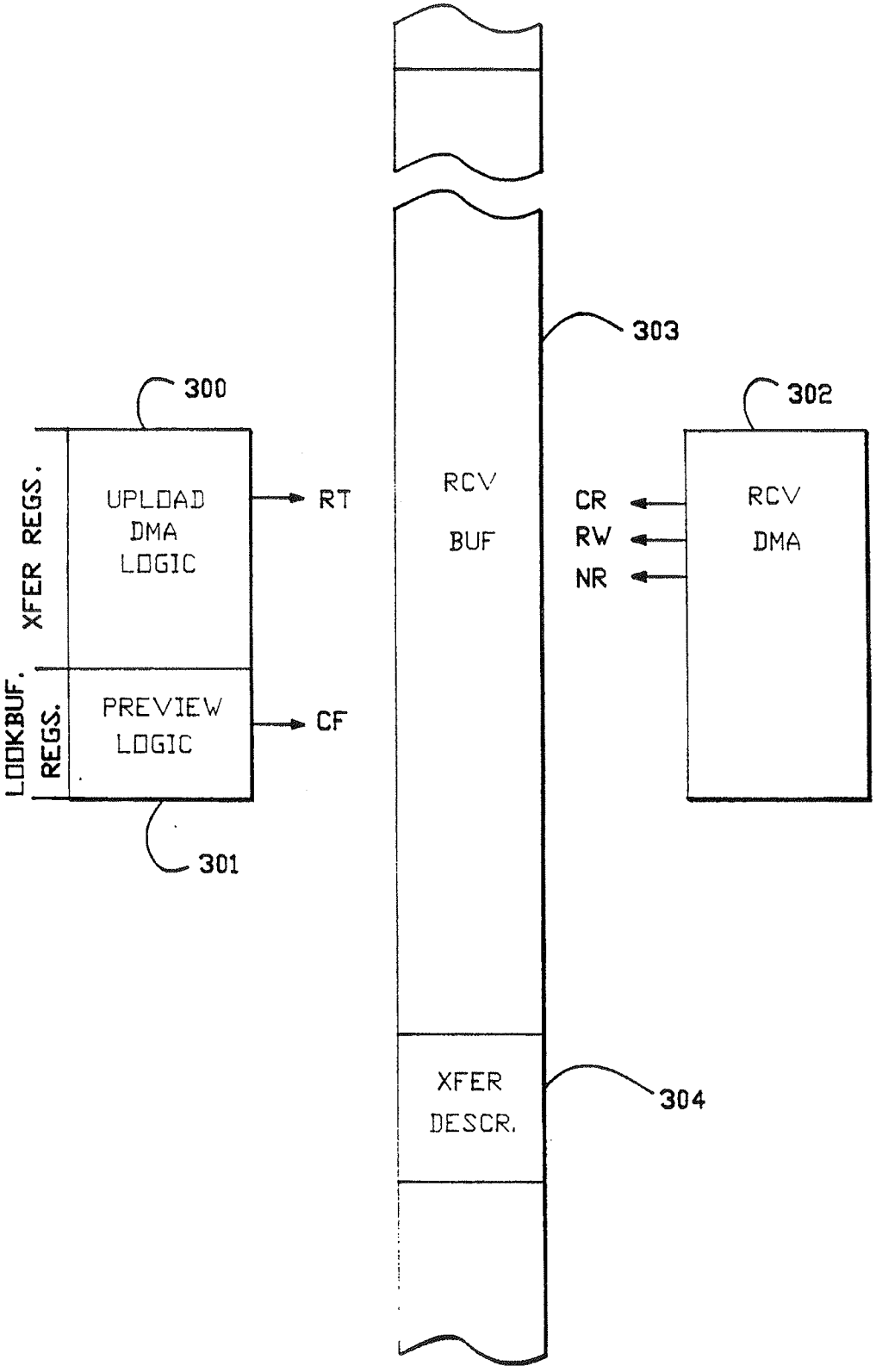

300
UPLOAD DMA LOGIC
XFER REGS.
RT
LOOKBUF. REGS.
PREVIEW LOGIC
CF
301
RCV BUF
303
XFER DESCR.
304
302
RCV DMA
CR
RW
NR

FIG.—11

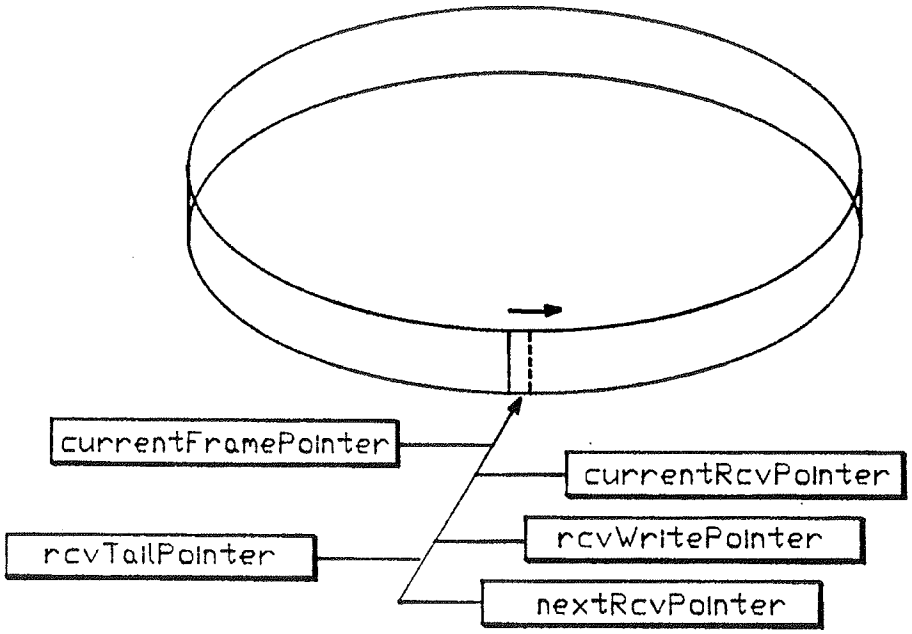


FIG.—12A

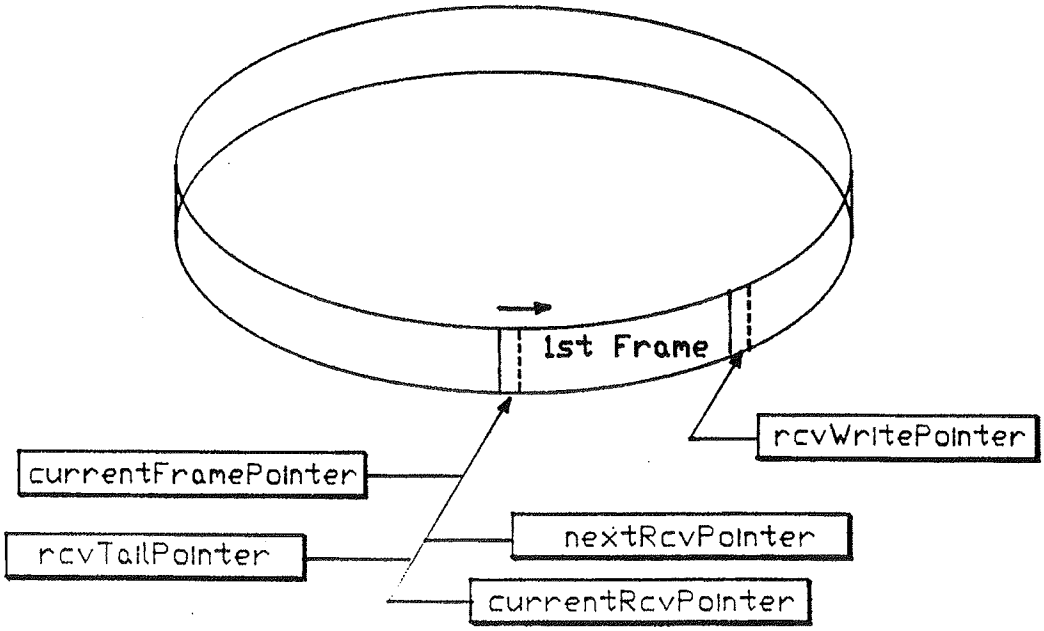


FIG.—12B

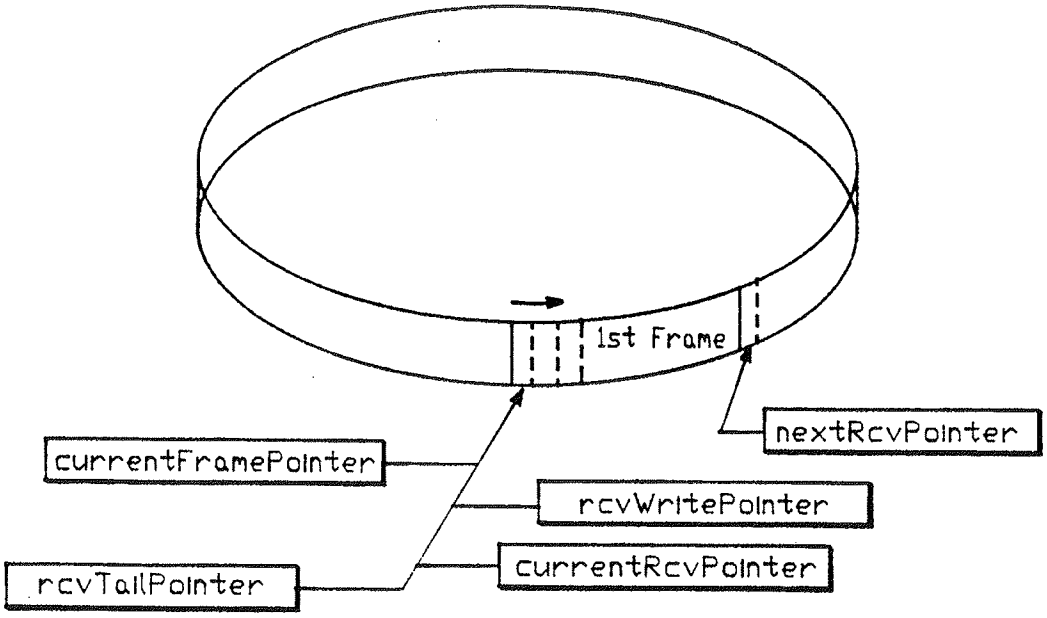


FIG.—12C

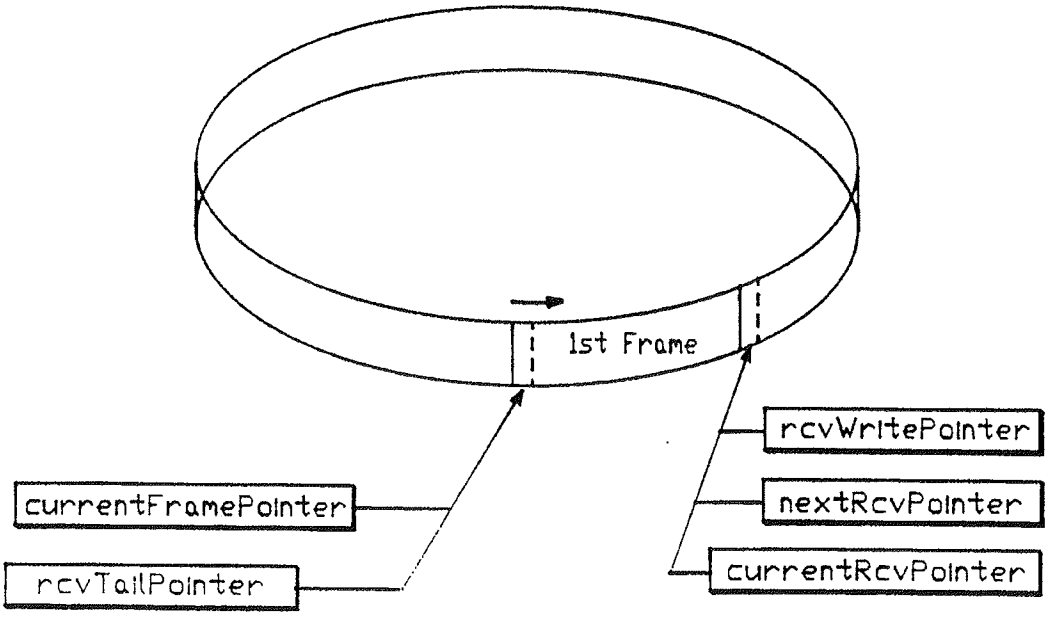


FIG.—12D

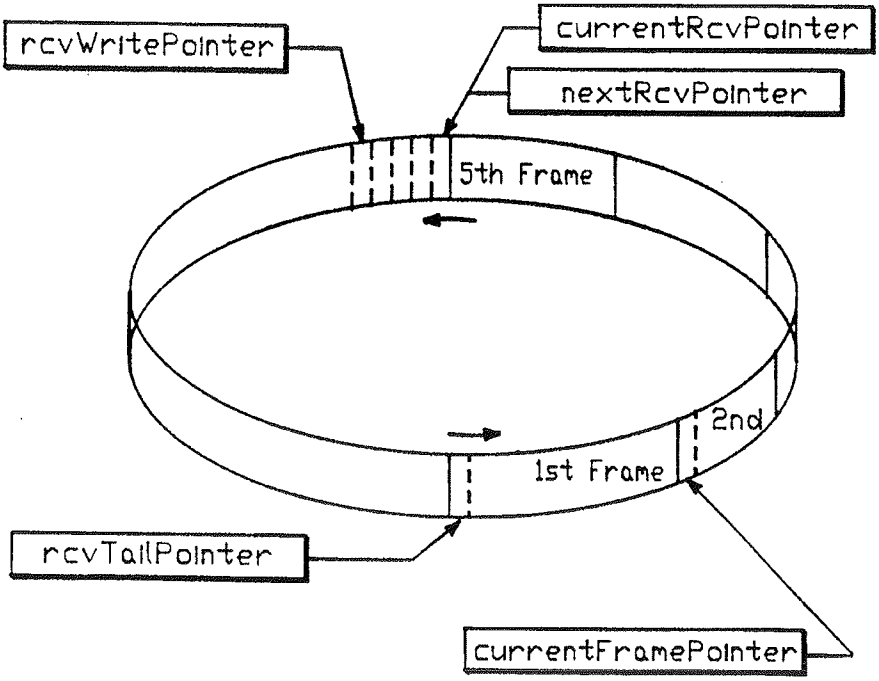


FIG.—12E

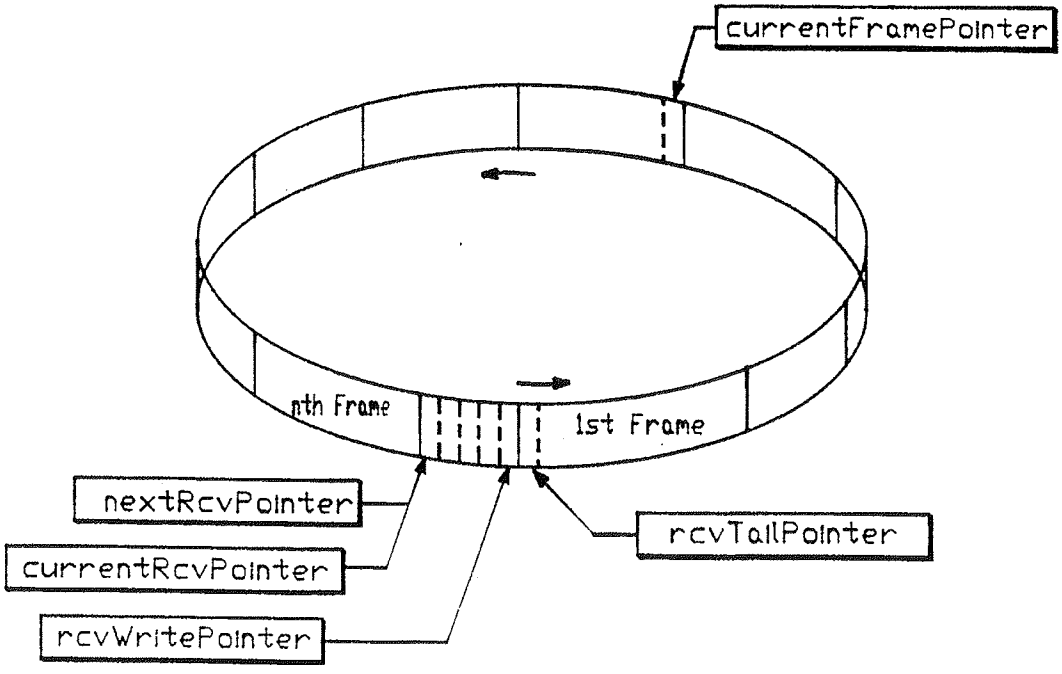


FIG.—12F

NETWORK INTERFACE WITH HOST INDEPENDENT BUFFER MANAGEMENT

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to architectures for network interface controllers, and particularly to architectures for managing the transmission and reception of data between a host system and a network.

2. Description of Related Art

Network interface controllers are involved in the management of the transfer of data between a host computer system and a communications network which is typically asynchronous and independent of the host system. Thus, data may be received across the network at times when the host is unable to process the received information directly. Similarly, when the host desires to transmit information on the network, the flow of data from the host system to the network must be managed. Prior art network interface controllers manage this asynchronous interface with the use of buffers in host managed memory. Received data is typically supplied from a media access controller (MAC) through a FIFO buffer on a network interface controller card into a host managed receive area buffer, either in the host address space or in adapter memory that is managed by the host, using DMA techniques that relieve the host processor from many of the sequential addressing tasks involved in uploading the data from the network into the receive area buffers. When the host desires to utilize or review the information received, h reads from the host receive area buffers, and transfers information out of the receive area buffers to permanent locations for use by the host system.

Similarly, when the host system transmits data, in prior art systems, it typically writes the data into a host managed transmit buffer area, and the network interface controller transfers that data from the host managed space using DMA techniques through a FIFO buffer in the interface controller and on to the network through a MAC.

Various devices have been developed which tend to optimize the DMA techniques used for managing the flow of data between the host system address space and the network interface controller. Representative prior art systems include the National Semiconductor DP83932B, a Systems-Oriented Network Interface Controller (SONIC) and the Intel 82586 Local Area Network Coprocessor. However, these prior art systems rely on host system address space for the receive and transmit buffers, and utilize a significant portion of the host processor overhead and host bus bandwidth.

It is desirable to provide a network interface controller which minimizes the use of host processor overhead and host system bus bandwidth, and simplifies the software executed by the host required for managing the interface.

SUMMARY OF THE INVENTION

The present invention provides a network interface controller which controls communication between a host system and a network transceiver coupled to a network which comprises a buffer memory outside of the host address space in which receive and transmit buffers are managed, host interface logic responsive to a prespecified range of host addresses, like memory mapped registers in the host address space, for mapping

data between the host address space and the buffer memory, and network interface logic coupled with the network transceiver, for mapping data between the buffers in the independent memory and the network transceiver.

Because the host interface logic and network interface logic manage accesses to the buffer memory, the host system is able to access the multiple data buffers for transmitting and receiving data through a limited prespecified address range. The dedicated memory mapped page in host address space is automatically remapped through the host interface logic into the buffer memory in operations that are transparent to the host.

According to one aspect of the invention, the buffer memory includes a transmit descriptor ring buffer storing a ring of transmit descriptors, the transmit descriptors including immediate data and data identifying data in the host system address space to be downloaded to the buffer memory for later transmission. The buffer memory also includes a transmit data buffer which receives data identified in transmit descriptors through the host interface logic during download, and supplies data to the network interface logic during transmission in response to transmit descriptors in the transmit descriptor ring buffer.

The buffer memory also includes a receive ring buffer and a transfer descriptor buffer. The receive ring buffer stores frames of data being received under control of the network interface logic. The transfer descriptor buffer stores a descriptor of locations in the host address space to which received data is to be uploaded through the host interface logic.

The host interface logic includes transmit descriptor logic, download logic, view logic, transfer descriptor logic, and upload logic in a preferred system. The transmit descriptor logic maps transmit descriptors from the host system to the transmit descriptor ring buffer. The download logic is responsive to transmit descriptors in the transmit descriptor ring buffer for retrieving data from memory in the host address space and storing the retrieved data in the transmit data buffer. The view logic presents data in the receive ring buffer to the host system prior to transfer of the data to memory in the host address space. The transfer descriptor logic maps transfer descriptors from the host system to the transfer descriptor buffer. Finally, the upload logic is responsive to the transfer descriptors in the transfer descriptor buffer, for transferring data from the receive ring buffer into memory in the host system.

The network interface logic includes receive logic and transmit logic. The receive logic moves data received from the network transceiver into the receive ring buffer. The transmit logic is responsive to transmit descriptors in the transmit descriptor ring, for retrieving data in the transmit descriptor ring buffer and the transmit data buffer, and supplying the retrieved data to the network transceiver for transmission on the network.

In a preferred aspect, the system includes a transmit ring pointer logic and receive ring pointer logic which operates independent of the host system. The transmit ring pointer logic maintains a plurality of pointers including:

(1) a host descriptor pointer indicating a location in the transmit descriptor ring at which the transmit descriptor logic stores a transmit descriptor being written

to a prespecified block of addresses from the host system,

(2) a download descriptor pointer indicating a location in the transmit descriptor ring of a transmit descriptor identifying a frame being downloaded by the download logic to the transmit data buffer, and

(3) a transmit descriptor pointer indicating a location in the transmit descriptor ring of a transmit descriptor identifying a frame being transmitted by the transmit logic on the network.

The receive ring pointer logic also maintains a plurality of pointers, including:

(1) a receive frame pointer indicating a location in the received ring to which data being received through the receive logic from the network is being written,

(2) an upload frame pointer indicating a location in the receive ring from which data is being uploaded by the upload logic to the host system in response to a transfer descriptor in the transfer descriptor buffer, and

(3) a host view pointer indicating a location in the receive ring at which data may be read through the view logic by the host system before uploading at a prespecified block of addresses.

Accordingly, an architecture for a network interface controller is provided which requires a much simplified driver software in the host system. Furthermore, the architecture reduces the bandwidth of the host system bus required for managing transmits and receives through a network.

Other aspects and advantages of the present invention can be seen upon review of the figures, the detailed description, and the claims which follow.

BRIEF DESCRIPTION OF THE FIGURES

FIG. **1** is a block diagram of a network interface controller according to the present invention.

FIG. **2** is a functional block diagram of the interface control chip shown in FIG. **1**.

FIG. **3** is a schematic diagram illustrating data flow from the host memory space through adapter memory to the network according to the present invention.

FIG. **4** is a map of host system address space used for the transmission and reception of data according to the present invention.

FIG. **5** is a memory map of the adapter memory independent of the host system address space.

FIG. **6** illustrates the transmit descriptor data structure according to one aspect of the present invention.

FIG. **7** illustrates the receive frame data structure.

FIG. **8** illustrates the transfer area descriptor format.

FIG. **9** illustrates the management of -the transmit descriptor ring buffer and transmit data buffer, and the pointers used during the transmit operation.

FIGS. **10A-10E** are a schematic illustration of the management of the pointers for the transmit descriptor ring buffer and the transmit data buffer.

FIG. **11** is a heuristic diagram illustrating the operation of the receive ring buffer and a transfer descriptor buffer, along with pointers generated by the host and network interfaces to manage these buffers.

FIGS. **12A-12F** illustrate the process of updating the pointers for the receive operation.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

A detailed description of preferred embodiments of the present invention is provided with respect to the figures. FIGS. **1** through **3** describe a representative system implemented according to the present invention. FIGS. **4** through **12A-12F** illustrate the data organization and flow in a preferred embodiment of the present invention.

I. System Overview

FIG. **1** is a schematic diagram of a computer system including the network interface controller according to the present invention. The computer system includes a host system, including a host processor **10**, host memory **11**, and other host devices **12**, all communicating through a host system bus **13**, such as an EISA bus. The host system bus **13** includes address lines which define a host system address space. Typically, for an EISA bus, there are 32 address lines establishing a host system address space of about 4 Gigabytes.

The network interface controller includes a network interface processor **14**, implemented in one preferred system as an application specific integrated circuit designed to implement the functions outlined below using VERILOG design tools as known in the art (available from Cadence, Inc, San Jose, Calif.). The network interface processor **14** is coupled through appropriate buffers to the bus **13**. The network interface processor **14** is also coupled to random access memory **15**, BIOS ROM **16**, and INFO EEPROM **17**, a test port **18**, an encode/-decode chip **19**, and a network transceiver **20**. The network transceiver **20** is, in turn, coupled to a network medium.

A majority of the functionality is embodied in the network interface processor **14**. In the preferred embodiment, all registers that are accessible across the bus **13** by the host system reside either in the processor **14**, or in the RAM **15**. If resident in the RAM **15**, their access is managed by the network interface processor **14**.

The RAM **15** is a primary resource on the network interface controller. This resource provides buffer memory outside the host address space used in the transmit and receive operations of the network interface. Details concerning the organization and utilization of this RAM **15** are described below.

The BIOS ROM **16** provides extension to the host system's basic input/output code through the network interface processor **14** during initialization. The addresses for the BIOS ROM **16** and the data from the BIOS ROM **16** are coupled to the network interface processor **14** across buses **21** and **22**, respectively, which are also shared by the RAM **15**.

The INFO EEPROM **17** stores critical adapter specific data used by drivers, diagnostics, and network management software. This data is stored during the manufacturing process. During initialization of the interface controller, the contents of the EEPROM **17** are loaded into a prespecified area of the RAM **15** for use during operation.

Coupled to the interface processor **16** is an encode/-decode chip **19**, such as the National Semiconductor 8391 Manchester encode/decode chip. The output signals **25** are coupled to a transceiver **20** and to an AUI connector **26** to allow use of a transceiver external to the board.

The transceiver **20** in a preferred system comprises either a thin Ethernet (coax/BNC) transceiver or a 10 BaseT (Type 3/RJ-45) transceiver. Control signals for the transceiver **20** are produced on the network interface controller **14**, using the conversion logic on the encode/decode chip **19**.

A test port **18** is provided in a preferred system for use during manufacture and testing.

II. Controller Functional Units

FIG. **2** provides a block diagram of the network interface processor **14** of FIG. **1**, including functional blocks and data paths. There are numerous connections not shown having to do with the control of the various data flow paths. The interfaces illustrated include a RAM interface **50**, a host bus interface **51**, and a transceiver interface **52**. The bus interface **51** is implemented for an EISA bus, and operates at times either as a master or as a slave on the bus. Each of the functional units in the implementation shown in FIG. **2** is described below.

A. EISA Slave Interface **5-4**

The EISA slave interface **54** provides a path for the EISA host bus to access the registers and buffers managed by the network interface controller. The module contains configuration registers for the controller, and performs crude decoding of the EISA bus for the purpose of routing signals. The EISA slave interface **54** does not interpret any of the addressing of individual registers distributed throughout the controller.

In operation, the EISA slave interface continuously monitors the EISA address bus and determines when the configuration registers, memory mapped network registers, or BIOS ROM of the adapter are being accessed.

In addition, for every memory slave cycle initiated by the EISA bus, the EISA slave interface will post a cycle request to the cycle arbiter **56**. The cycle arbiter imposes wait states upon the host system until the request has been granted.

The EISA slave interface also provides a generic 32 bit bus interface to the remainder of the network controller. The generic nature of the interface allows for easy adaptation of the design to other bus types, such as the microchannel, without requiring redesign of the remainder of the chip.

Bits **14-2** of the EISA address bus are latched and pass through to other modules. The least significant two bits of the address are represented by 4 byte enables that are also valid throughout a data transfer cycle. Four 8-bit byte lanes make up the slave data channel. The data bus is actually a pair of unidirectional buses, one for writes and one for reads in a preferred system. The data write bus is wired in a multi-drop fashion to all modules that require connection to the EISA data bus through the slave interface. The read bus is multiplexed and masked in the RAM interface module **50**. Write requests by the EISA bus can be held until they are acknowledged by the cycle arbiter **56**. When a cycle is held, the EISA bus may be released from wait states while the cycle completes on the adapter. If a second cycle is generated by the EISA bus while the first one is still pending, then the EISA bus will be held off with wait states until the pending write is performed. In this specific embodiment, pipelining of EISA reads is not supported.

The EISA slave interface also provides an interface to the EEPROM **17**. This interface operates to transfer the contents of the EEPROM **17** into the adapter memory after reset.

There are numerous registers in the EISA slave interface module **54**, primarily related to configuration of the adapter that conform to the EISA bus specification. These registers do such things as set up the adapter's memory base address, the interrupt level, the transceiver type selection, and the BIOS ROM enable. The configuration registers also provide the host with a positive means of identifying the adapter type and to globally disable the adapter.

B. EISA Master Interface **55**

The EISA master interface **55** handles requests from the upload DMA **57** and download DMA **58** for performing bus master operations across the EISA bus. The EISA master interface **55** autonomously arbitrates between pending upload and download requests, because of the EISA bus disallowing mixed reads and writes while performing burst transfers, used by the DMA operations of the preferred embodiment.

The bus master transfers are always initiated by either the upload DMA **57** or the download DMA **58**. The transfers may be terminated by either the DMA modules upon completion of a transfer, or by the EISA master interface upon preemption by another arbitrary device on the EISA bus.

Thus, the function of the EISA master interface **55** is to arbitrate for access to the EISA bus when transfer requests are pending from either or both of the upload DMA **57** and the download DMA **58**. The EISA master interface **55** performs the signalling necessary to establish first transfers with address slaves on the bus. It also ensures compliance with the EISA bus definitions.

This module also converts real mode segment:offset addresses to 20 bit linear addresses when enabled by the adapter mode logic **59**.

C. Master/Slave Union Module **53**

The master/slave union module **53** provides for sharing of connections to the EISA bus by the EISA master interface **55** and the EISA slave interface **54**. This union module **53** consists primarily of a series of 2:1 multiplexers.

D. Interrupt Controller Module **60**

The controller also includes an interrupt controller module **60**. The interrupt controller module **60** implements various interrupt and indication functions, including masking and enable/disable mechanisms. Interrupt signals are generated by various modules within the controller, and are routed to the interrupt controller module **60**. The interrupt controller module **60** then passes the interrupt signals through various enables and masks before OR-ing them together and driving the result onto the host bus.

The interrupt controller module **60** does not detect interrupt worthy events or acknowledge the interrupts passed to the host. It includes a number of access resident registers utilized in the interrupt handling functions.

E. Adapter Mode Module **59**

The adapter mode module **59** provides a number of functions including setting various basic operating modes of the controller, and reporting status of various conditions of the controller. The adapter module **59** also establishes the base address of a window register used for diagnostics by the host system. Furthermore, the adapter mode module **59** generates reset functions for the adapter. Also, this module provides the MAC I D register which identifies the media access controller implemented by the device, for communication to various modules within the controller and to the host system.

F. Cycle Arbiter Module **56**

The cycle arbiter module **56** is responsible for distributing access to the adapter's RAM resident and ASIC resident registers through the RAM interface **50** among various requestors. It functions to allow timely access to

the RAM by modules that are most in danger of suffering an overrun or underrun condition in response to a priority scheme.

G. Multicast Comparator Module 61

The controller illustrated in FIG. 2 also includes a multicast comparator module 61. When enabled by the adapter mode module 59, the multicast comparator module 61 performs a bit by bit comparison of a received frame's destination address field with contents of the multicast address table. The multicast address table is established by the host and stored in RAM 15. A mismatch during this compare, coupled with neither an individual address nor a broadcast address match, will result in the rejection of an incoming frame.

Thus, the multicast comparator module 61 monitors the activity of the Ethernet receiver module 62 and the receive DMA module 63 to determine when a new frame is being received. Each byte that is received by the Ethernet receiver 62 and presented at the parallel interface 64 of the receiver, is shadowed by the multicast comparator module 61. These bytes are then compared against valid entries in a multicast address table accessible by the multicast comparator 61.

The multicast comparator 61 does not establish or maintain the contents of the multicast address table. However, the module detects the host accesses to the table and supplies appropriate redirection offsets to the RAM interface module 50.

H. Statistics Controller Module 65

The preferred system also includes a statistics controller module 65. This module monitors activity of various other modules within the controller, most particularly the Ethernet transmitter module 66 and the Ethernet receiver module 62, and updates statistics maintained in RAM 15 as applicable events occur.

I. Download DMA Module 58

The download DMA module 58 is responsible for issuing requests for bus master downloads of data from the host system to the adapter memory. This data is then deposited within the adapter's onboard transmit data buffer for either immediate or future transmission.

As soon as buffer descriptors are defined and one of the transmit data buffers becomes available, as described below, the download DMA module submits requests for download bus master operations to the EISA master interface 55. The download DMA module 58 performs byte alignment, including any required packing and unpacking to align the data as implied by the respective starting addresses of the host and the adapter.

The download DMA module 58 also includes logic for maintaining the transmit descriptor ring buffer within the adapter's RAM. The download DMA module 58 generates an interrupt in the appropriate mode to indicate completion of the download operation. Also, the download DMA module 58 informs the transmit DMA module 67 when it is time to begin transmission. Various registers involved in the download DMA operation are described in more detail below as they are involved directly in the data buffering process of the present invention.

J. Transmit D A Module 67

The transmit DMA module 67 operates to fetch bytes from the transmit descriptor buffer, the transmit data buffer, or both, as described below, when directed to do so by the download DMA logic 58. The fetched bytes are then presented in sequence to the Ethernet transmitter module 66.

Thus, the transmit DMA module 67 functions to read the contents of the transmit descriptor ring buffer to determine the extent of immediate data, and the length of the overall frame to be transmitted. If a frame specified does not match the specifications for the network, for instance, is shorter than the 802.3 minimum, and the adapter is so enabled, this module will supply additional bytes of undefined data to the Ethernet transmitter module 66 as padding.

Collision retries in the CSMA/CD network are handled by the transmit DMA module 67. When a collision is indicated by the Ethernet transmitter 66, the transmit DMA module 67 will resubmit the same frame by reinterpreting the frame descriptor in the buffer.

If enabled, and when the transmit complete condition is satisfied, a transmit complete interrupt is generated for handling by the interrupt controller 60 in the transmit DMA module 67.

The transmit DMA module 67 also stores appropriate frame status in the transmit descriptor ring buffer of the adapter upon completion of transmission.

The transmit DMA module 67 also detects underrun conditions, when a shortfall of data available for transmission occurs.

Again, registers involved in operation of the transmit DMA module 67 are described in detail below.

K. Ethernet Transmitter Module 66

The Ethernet transmitter module 66 is an essential implementation for an 802.3 standard network. This module accepts parallel data bytes from the transmit DMA module 67 and applies the 802.3 access rules, and supplies serial data to the external encoder/decoder chip.

L. Ethernet Receiver Module 62

Similarly, the Ethernet receiver module 62 is an essential 802.3 implementation. This module accepts serial data from an external encoder/decoder, applies the 802.3 rules to the data and presents the data in parallel form for use by the receive DMA module 63. Thus, the Ethernet transmitter 66 and Ethernet receivers perform the standard CSMA/CD functions.

M. Receive DMA Module 63

The receive DMA module 63 is a complement function to the transmit DMA module 67. This module is responsible for accepting parallel data bytes on the Ethernet receiver 62 and depositing them in the adapter's receive ring buffer.

The receive DMA module 63 is responsible for assembling bytes from the Ethernet receiver into 32 bit words prior to storage in the adapter's RAM. At completion of frame reception, the frame's status and length are deposited within the receive ring buffer for use by the host system.

The receive DMA module 63 is also responsible for establishing and maintaining of the receive buffer ring within the RAM of the adapter as described in detail below. Furthermore, the positioning of the "LOOKBUF" register allowing the host to view received data, as described below, is handled by the receive DMA module 63.

The receive DMA module 63 also handles interrupt indications under appropriate conditions.

N. Upload DMA Module 57

The upload DMA module 57 performs data transfers from the receive buffer through the RAM interface 50 to the host system. Thus, the receive ring buffer is managed by the receive DMA module 63 and interpreted by

the upload DMA **57**. Actual bus master transfers are carried out by the EISA master interface module **55**.

The upload DMA module **67** interprets data structures deposited in the receive ring buffer by the receive DMA module **63** including the location and length of a received frame. It also reads the transfer descriptor which is defined by the host system to determine how many bytes of the frame to transfer, and where in the host memory to transfer the frame.

The upload DMA module **57** requests bus master cycles from the EISA master interface module **55** in response to valid transfer requests deposited by the host.

The upload DMA module **57** also utilizes an interlock between the receive DMA module **63** and the upload DMA module **57**, to reduce the transfer rate to the host system to prevent "getting ahead" of the frame being received through the receive DMA module **63**. Finally, this module generates interrupts indicating completion of a transfer for use by the host. Various registers involved in this procedure are described below.

O. RAM Interface Module **50**

The RAM interface module **50** provides multiplexers and masks for various functions involved in addressing the RAM. The module multiplexes the various address and data sources together to form parameters for RAM access cycles. This module **50** is responsible for gathering up data from the various other modules on the controller that can be read by the host system. Additionally, this module applies a mask to the data to force unused upper bits to a zero, and latches data words for multicycle reads.

P. JTAG Module

Also included in the controller, though not shown, is a JTAG module which implements a state machine as specified in IEEE standard 1149.1-1990, May 21, 1990. This module provides a scan test of the ASICs pins for use during manufacture.

III. Transmit and Receive Data Flow and Structure

FIG. **3** provides a heuristic data flow diagram of an adapter according to the present invention emphasizing the host interface, the adapter memory and the network interface data flow for transmission and reception.

As mentioned above, the host system will include a host memory space (generally **100**) defined by the addresses on the host bus. A pre-specified block **101** of the host memory space is set aside for the adapter interface addresses. The adapter includes host interface logic **102** which is responsive to accesses across the host bus within the adapter interface address block **101**. Also in the adapter is a host independent memory **103**. The host interface logic operates the transfer data between the specified block **101** of addresses and the independent memory. The adapter also includes network interface logic **104** which is coupled to the adapter memory. The network interface logic manages transfers of data from buffers in the independent memory **103** and the network transceiver **105**. The network transceiver **105** then supplies the data onto the network medium **106**.

The host interface logic includes a transmit descriptor logic and a download DMA logic (generally **107**) used in the transmit process, and view logic, transfer descriptor logic, and upload DMA logic (generally **108**) used in the receive process. These modules basically manage communication of data between the independent memory **103** and the host in response to writes by the host system to the adapter interface address block **101**. This relieves the host of any address translations or buffer management functions for the transmit and receive operations.

The network interface logic **104** includes transmit DMA logic, (generally **109**) and receive DMA logic (generally **110**). The transmit DMA logic **109** is responsive to descriptors stored in the adapter memory **103**, as described below, for moving data out of the independent adapter memory **103** to the network transceiver **105**. Similarly, the receive DMA logic **110** is responsible for moving data from the transceiver **105** into the independent adapter memory **103**. Thus, all communications of data from the network medium **106** are coupled directly into host independent memory **103**. Communications from the host independent memory **103** are then controlled through the host interface logic **102** in response to a memory mapped region in the host memory space, greatly simplifying the protocol software necessary to communicate with the network.

FIG. **4** provides a simplified map of the adapter interface host address block **101**. The addresses within this block appear to the host like memory mapped registers in a continuous 8K block of the host address space in a preferred system.

For the EISA embodiment, "registers", or mapped areas, in the block **101** are located on double word address boundaries, thus, addresses are a multiple of four. Many of the "registers" occupy several double words (as many as **509**) of memory space.

Although the "registers" are memory mapped to an arbitrary prespecified block of host address space, none of the reads or writes performed by the host system to these registers actually directly access the adapter memory. Rather, the accesses to the memory mapped space are interpreted by the host interface logic **104** transparent to the host system. Thus, the memory in the adapter is independent of the host address space and of host management. FIG. **4** provides an overview mapping of the adapter interface host address space used for accessing these registers. The registers include primarily a transmit area register (XMIT AREA) at offset 0010 (hex), a transfer area register (XFER AREA) at offset 0800 (hex), and a look buffer (LOOKBUF) at offset 100C (hex). Various status, statistics, information, and indication registers are distributed throughout the balance of the area.

The XMIT AREA register is used by the host to write transmit descriptors into the adapter. The transmit descriptors are described in more detail below, but include data that identifies data to be compiled and transmitted as a frame, and may include immediate data. The XMIT AREA at offset 0010 (hex) is approximately 2K bytes in size. This data is mapped into a transmit descriptor ring in the independent adapter memory as described below.

The XFER AREA at offset 0800 (hex) in the adapter interface host address block is a buffer of approximately 1K byte through which transfer descriptors are written into the independent memory of the adapter. The LOOKBUF area at offset 100C (hex) is a buffer of approximately 2K bytes providing a read only window into a receive ring buffer within the host independent adapter memory.

FIG. **5** provides a map of the host independent memory on the adapter. This memory is organized into a transmit data buffer at offset 0 of approximately 3K bytes, a transmit descriptor ring at offset 0C00 (hex) of approximately 5K bytes, a receive buffer ring at offset 2000 (hex) of approximately 22K bytes, and a transfer

descriptor area at offset 7800 (hex) of approximately 1K bytes is provided in the independent memory. The last three areas of memory include adapter information, network statistics, and multicast address tables for use by the adapter.

In the preferred system, the adapter uses 32K bytes of static RAM for the transmit buffers, receive buffers, control structures, and various status and statistics registers. Several of the regions in the adapter's memory defined in FIG. 5 provide defined data structures.

A. Transmit Data Buffer

The transmit data buffer occupies 3K bytes as mentioned above. This region is divided into two 1.5K buffers. Only the data that are downloaded to the adapter via bus master transfers are stored in these buffers. The controller will use both the contents of the transmit data buffer and the immediate data portion of the transmit descriptors, when encapsulating a frame for transmission. The adapter automatically alternates the use of the buffers after choosing the buffer closest to the base of the memory as the power up default.

The transmit buffers are shared by the download DMA logic and the transmit DMA logic. The transmit DMA logic may switch from buffer 0 to buffer 1 and back again freely. The only restriction being the availability of transmit data as defined by the transmit start threshold register. The transmit DMA module switches from one buffer to the other whenever it has completed a transmission. The buffer switch occurs regardless of whether or not the transmission was successful and regardless of whether or not bus master download data were used in the preceding transmission.

The download DMA module may only switch from one buffer to the other, if the buffer it is going to switch to is not being used by the transmit DMA module. Download DMA will attempt to switch from one buffer to another every time it completes processing of a transmit descriptor as described below, regardless of whether or not any bus master operations were called for in the preceding descriptor. However, it will not change to a buffer that is in use by the transmit DMA module.

B. Transmit Descriptors

Transmit descriptors define frames that are pending transmission, and hold the status of frames that have been transmitted. These descriptors are of variable length and are arranged in a sequential fashion around a 5K byte ring buffer as mentioned above. The first entry of the descriptor must align on a double word boundary. FIG. 6 illustrates the transmit descriptor data structure.

The bulk of the contents of an entry into the transmit descriptors region is copied verbatim from the data supplied by the host processor via the XMIT AREA illustrated in FIG. 4. However, in order to comply with the format requirements of the XMIT PROT ID and XMIT FRAME STATUS registers and to supply sufficient information for frame transmission and buffer management; one value must be relocated and several must be deposited automatically.

The host processor's writes to the XMIT AREA are offset automatically by the adapter such that the first value written—XMIT PROT ID and XMIT REO HANDLE—end up in the fifth 32 bit word location (offset 10 (hex)) in the next available data structure in the ring. This means that the XMIT REQ HANDLE value is written by the host to the location reserved for the MACID value. Immediately after the XMIT REQ HANDLE value is written to the adapter's RAM, the adapter must copy the contents of the least significant 16 bits of the fifth 32 bit word location to the most significant 16 bits of the fourth 32 bit word location. After copying XMIT REO HANDLE, the adapter will retrieve the MACID value from the MACID register and write that to the location vacated by XMIT REQ HANDLE. Later, after frame transmission, the least significant 16 bits of the fourth 32 bit location (offset C (hex)) will be updated with the transmit frame's status.

The NEXT DESCRIPTOR POINTER entry may be updated by the adapter any time after the XMIT BUFFER COUNT and XMIT IMMED LEN values have been written to the adapter. Because the data written to the XMIT AREA register must be written in a precise order to exact locations, the writes of these two values can be easily detected and used for the descriptor size calculations required to determine the start of the next descriptor without having to retrieve the values from RAM.

Finally, once the last XMIT DATA LEN value has been written to the adapter, the frame length can be calculated and deposited in the FRAME LENGTH position of the data structure. This value is also copied to the XMIT FRAME LENGTH register in the controller chip for immediate use by the host.

The next several paragraphs define each of the fields of the transmit descriptor data structure.

The NEXT DESCRIPTOR POINTER value points to the first word of the next descriptor. This value is updated immediately after the host reads XMIT QUEUE STATUS. NEXT DESCRIPTOR POINTER being defined does not necessarily imply that the location pointed to contains a valid descriptor. It merely indicates where the next valid descriptor may be found once it is defined.

The FRAME LENGTH field is computed and updated by the adapter. The frame length is calculated by summing all of the XMIT DATA LEN values and the XMIT IMMED LEN value. The resulting sum is the total number of bytes in the transmit frame. If the sum is less than the 802.3 minimum frame length, then the sum will be set to the minimum frame length value. The sum is written to the FRAME LENGTH line of the transmit descriptor and is also made available to the host via the XMIT FRAME LENGTH register.

The XMIT FAILURE field contains the error code that is made up of the status bits gathered from the Ethernet transmitter after the completion of transmission. This field is mapped to the XMIT FAILURE register for host access.

The XMIT REQ HANDLE value is interpreted by the transmit DMA controller to determine whether or not to generate an indication upon completion of the transmission attempt(s) for the associated frame. If the field is non-zero, an indication will be generated. Also, the frame's entry in the transmit descriptor ring will be maintained until the host has had an opportunity to examine the transmit status. The XMIT REQ HANDLE, XMIT STATUS, XMIT PROT ID, and the MACID fields are all made available to the host when an indication is generated. If XMIT REQ HANDLE is a zero, then the transmit descriptor queue entry is discarded after transmission without notifying the host in any way. Transmit underrun conditions are posted regardless of a zero XMIT REQ HANDLE.

The XMIT STATUS field contains the transmit status for the associated frame. The contents of this field

are updated immediately after the transmission attempt(s). The return codes are defined in the XMIT STATUS register definition.

The XMIT PROT ID value in this field is merely maintained within the queue for use by the host upon completion of the transmission to identify the particular protocol responsible for the frame. It allows simultaneous activity of multiple protocols. Together XMIT PROT ID and XMIT REQ HANDLE uniquely identify a frame passing through the adapter.

The MACID, like XMIT PROT ID, is maintained within the queue for use upon completion of the transmission. However, the host does not write this value to the adapter via the XMIT AREA register. Rather, the host stores this value once in the MACID register and then depends upon the adapter to deposit this value into the descriptor data structure after XMIT REQ HANDLE has been copied to its final position.

The contents of the XMIT BUFFER COUNT field are supplied by the host via a write to XMIT AREA. This field specifies the number of buffers within the host's memory that are used to make up the transmit frame. Each of the buffers is transferred in the order listed from the host's memory to the adapter's transmit data buffer as soon as one of the two transmit data buffers becomes available. If XMIT BUFFER COUNT is zero, then no bus master operations will be performed for this frame.

The XMIT IMMED LEN field, defined by a write from the host to XMIT AREA, specifies the number of "immediate" data bytes that will be supplied by the host. If this field is zero, then the next 32 bit word location will contain the first host data buffer descriptor and the entire frame will be transferred to the adapter via bus master cycles. The XMIT IMMED LEN value will not necessarily be a multiple of four. The location of the first host data buffer descriptor is determined as follows:

```
DESCRIPTOR OFFSET (hex)=((XMIT IMMED
  LEN+3)& fffc)+18;
```

The variable length IMMEDIATE DATA field contains the immediate data deposited to the adapter by the host using memory writes to XMIT AREA. This field may vary in length from 0 to 1,514 bytes. Immediate data is inserted into a transmit frame between the preamble and the transmit buffer data (if any) by the adapter during transmission. Generally, immediate data is made up of the destination and source addresses and any protocol-specific header data. It is reasonable, however, for the entire transmit frame to be considered immediate data. This would make it unnecessary for the adapter to perform any bus master operations to fetch the remainder of the transmit frame. If XMIT IMMED LEN is zero, then this field is skipped and the entire frame is assumed to reside in host memory resident data buffers. If XMIT IMMED LEN does not specify an integral number of double words, then the host may round up to the nearest multiple of 4 and write up to that number of bytes. The extra bytes, beyond XMIT IMMED LEN, will be ignored and not included as part of the transmitted frame.

The XMIT DATA LEN field, one of two entries per host data buffer descriptor, defines the number of bytes in the associated host buffer. This value need not be a multiple of four.

The 32 bit XMIT DATA PTR value is the physical starting address of the associated host data buffer. This value need not be a multiple of four.

C. Receive Buffer

The receive buffer is a 22K byte ring of variable length receive frames. Each frame is preceded by a header that defines the frame's size and status and the location of the header of the next frame in the ring buffer.

The beginning of a receive frame entry in the ring can begin on any 32 bit word boundary. The receive data is buffered in the format illustrated in FIG. 7.

The 32 bit NEXT RCV PTR value contains the address of the NEXT RCV PTR value of the next entry in the ring. This value becomes valid upon completion of the reception of the present associated frame. The buffer pointed to by NEXT RCV PTR may not necessarily contain a valid frame. This must be determined by the ring maintenance pointers that determine the beginning and end of the valid entries within the ring.

The RCV FRAME STATUS word contains the various error codes regarding the condition of the associate frame. RCV FRAME STATUS is updated immediately after frame reception. The contents of this entry are made available to the host via the RCV FRAME STATUS register.

The upper 16 bit word of the RCV FRAME SIZE entry is occupied by the adapter's MACID value. This value is retrieved from the MACID register and deposited in this word of the receive buffer data structure at the same time that the RCV FRAME SIZE value is posted.

The length of the received frame is deposited in the RCV FRAME SIZE register immediately after the frame has been received. The contents of this entry are made available to the host via the RCV FRAME SIZE register.

The RECEIVE DATA field varies in length from 1 to 1,514 bytes. The receive frame—starting with the destination address—is stored in this field as the frame is being received.

D. Transfer Descriptor

The adapter in one preferred implementation accommodates one and only one transistor descriptor at a time. Attempts to download a second descriptor while the first one is being processed will result in a return code from the XFER QUEUE STATUS register that indicates a lack of resources to accept the request. Only the frame currently visible via the LOOKBUF is acted upon by the transfer specification within the XFER AREA register.

The transfer descriptor is stored in adapter RAM using the format of FIG. 8.

The TD OFFSET word defines the offset into the received frame from which the transfer will commence. This value need not be a multiple of four. Data that the host has already examined (via the LOOKBUF) may not need to be transferred to the host. Therefore, a non-zero value written into the TD OFFSET field will cause the bus master circuit to offset its start address by TD OFFSET bytes before beginning the transfer. This value may range from zero (transfer from the beginning of the frame) to 1,514.

The number of length/pointer pairs is defined in the TD BUFFER COUNT field. As many a 127 entries are allowed. Beyond that, XFER AREA buffer exhaustion is assured. This value must be non-zero.

The TD DATA LEN field contains the size of the host's receive buffer into which receive data will be transferred. The adapter is not restricted to transferring only 32 bit words. Attempting to transfer more than the maximum frame length of 1,514 bytes will cause the adapter to terminate the transfer upon reaching the end of the frame.

The 32 bit TD DATA PTR value is the physical address of the first byte of the host's receive buffer. This value need not be a multiple of four.

The physical address of the data buffer in the host to which a received frame will be uploaded is written to the adapter as a 32 bit TD DATA PTR value in XFER AREA. The adapter will use this value as a pointer to the first byte in the receive buffer. There are no restrictions placed on the value of this pointer by the adapter. Byte, word and double word alignment of the buffer data are all supported.

E. Adapter Info, Network Statistics, Multicast Address Tables

The adapter info, network statistics, and multicast address tables in the adapter RAM memory are utilized for various functions by the adapter, as will be understood by those skilled in the art, not directly relevant to the transmit and receive operation subject of the present application.

IV. Transmission Process

FIG. **9** illustrates the network interface logic and host interface logic used in managing the transmit data buffer and transmit descriptor ring buffer in the independent memory on the adapter. On the host interface side, the logic includes host descriptor logic **150** and download DMA logic **151**. The host descriptor logic **150** and download DMA logic **151** are coupled to the host address space through the transmit "registers" including the XMIT AREA register, the XMIT COMPLETE THRESH register, the XMIT FAILURE register, the XMIT FRAME LENGTH register, the XMIT FRAME STATUS register, the XMIT PROTID register, the XMIT QUEUE STATUS register, and the XMIT START THRESH register. Details of these registers are described below.

The descriptors illustrated in FIG. **6** are stored in the transmit descriptor area of the host independent RAM on the adapter by host writes to the XMIT AREA address block. Three different processes act upon entries in the descriptor queue. The host writes to initially create transmit descriptors, bus master downloads to move buffer data from host memory to the transmit data buffer and transmission of described frames on the network. The first two processes occur within the download DMA logic **151** and the host descriptor logic **150**. Transmission is performed by the transmit DMA logic **155**. During the course of adapter operations, the number and status of descriptors in the transmit descriptor ring buffer **152** will vary according to the relative speeds of the host write, download, and transmission processes.

Two variables within the download DMA logic **151** helped describe the status of the transmit descriptor queue. ZERO DOWNLOADS PENDING indicates that there are no complete frame descriptors yet to be processed by the download DMA logic **151** for download. The ZERO FRAMES RESIDENT variable indicates that there are no descriptors which have been already processed by the download process, but are yet to be transmitted.

Each of the three processes that make up the transmission mechanism maintains its own set of pointers to the descriptor queue.

The host descriptor logic **150** generates pointers for the transmit descriptor ring buffer **152** on the adapter memory, which identify the current position of host accesses for writing descriptors in the transmit descriptor ring buffer **152**. These pointers are designated the current host descriptor pointer CHD, and the host write pointer HW. The current host descriptor pointer CHD points to the base address of a descriptor currently being written, or expected to be written by the host. The host write pointer HW points to the location within the current descriptor (pointed to by the CHD pointer) to which the host is expected to write next. That is, the HW pointer predicts the offset within the XMIT AREA address block at which the host will be writing to complete the current descriptor.

The download DMA logic **151** generates **3** pointers, and a buffer select signal, while managing downloads from the host system into the transmit data buffers XMIT DATA BUFO **153** and XMIT DATA BUF1 **154**. The pointers generated by the download DMA logic **151** include the current download descriptor pointer CDD which points to the base address of a descriptor that is currently being processed by the bus master download DMA logic **151**. The second pointer generated by the download DMA logic includes the download data pointer DD which points to the location within the current transmit data buffer (either XMIT DATA BUFO or XMIT DATA BUF1) to which the download process is writing data. The third pointer generated by the download DMA logic **151** includes the current download buffer CDB pointer. The CDB pointer points to the buffer descriptor within the transmit descriptor pointed to by the CDD pointer, in which the specification of the buffer in host memory subject of a current download process resides.

The download DMA logic also selects the current transmit data buffer **153** and **154** to which the download DMA logic transfers data in the bus master operation as heuristically illustrated by signal BUF1/0.

The transmit DMA logic **155** generates three pointers for the transmission process. These pointers include the current transmit descriptor CXD pointer, which points to the base address of the descriptor in the transmit descriptor ring buffer **152** currently being processed by the transmit logic **155**. The transmit read XR pointer indicates the location within the current descriptor or current transmit data buffer (**153** or **154**) from which the transmission process is reading data to be transmitted.

The XMIT TAIL pointer (XT) points to the back end of the queue. The XT pointer points to an older descriptor in the transmit descriptor ring buffer **152** than the CXD pointer when there are frames that have completed transmission, but have not yet had their status information read by the host system.

The preferred system operates in two modes relative to generating indications to the host of the completion of a transmission. The preceding paragraphs described the data structures associated with host write, bus master download, and transmission processes when a XMIT COMPLETE ON DOWNLOAD variable is false and the indicator is generated when the transmission is complete, or when XMIT COMPLETE THRESH (described below) is met. When XMIT COMPLETE ON DOWNLOAD is true, the download DMA logic **151** is also responsible for keeping

track of the frames for which status has not been read. In this mode, the host receives the transmit complete indication for a frame upon download of the frame so it is possible that the frame can be transmitted before the host is able to respond to the indication. This condition allows the XMIT TAIL pointer to no longer define the oldest useful entry in the descriptor ring buffer. Therefore, a frame status process is necessary. The frame status pointer FS points to the base address of the oldest descriptor for which status has not yet been read by the host, in this mode of operation. Since the FS pointer is related to the transmit process, it is illustrated heuristically in the transmit DMA logic **155** in FIG. **9**. In the preferred system, however, it is logically coupled with the download DMA logic **151**.

The transmit DMA also selects the current transmit data buffer **153** or **154** from which a current transmit frame is being composed, as indicated heuristically by the signal BUFO/1.

The interface to the adapter is seen by the host as if it were a set of registers at a prespecified address block. The significant "registers" involved in transmission are outlined below.

A. XMIT AREA

The purpose of this register is to provide a mechanism for the host to supply the adapter with immediate data and a data structure that defines where in the host's memory system the remainder of the frame (if any) resides. The adapter stores this information in the transmit descriptor ring **152** for use when the frame being described can eventually be transmitted.

The adapter uses the address to which the data is written and the order in which k is written to determine what the data represents. The data must be written to the adapter using the structure and order described above with respect to FIG. **6**.

Bus master downloads begin after a descriptor has been written to XMIT AREA and XMIT QUEUE STATUS (described below) has been read by the host.

Once the host has completed the transfer of the transmit descriptor buffer structures to XMIT AREA, the host may read XMIT FRAME LENGTH to determine the number of bytes that the host has specified to be included in the transmit frame. XMIT QUEUE STATUS should then be read to advance the CHD pointer so that another frame may be written to this register. After reading XMIT QUEUE STATUS, the contents of XMIT FRAME LENGTH are undefined, until XMIT AREA is filled again.

The actual frame transmission onto the network will commence when two conditions are met: (1) the XMIT START THRESH (described below) condition has been met, or, if XMIT START THRESH is zero, when the entire frame has been copied to the adapter's RAM, and (2) when there are no previously queued transmit requests. If more than XMIT START THRESH bytes of immediate data are written to XMIT AREA, then network transmission may begin before XMIT QUEUE STATUS is read.

If the adapter runs out of XMIT AREA resources while the host is writing data to XMIT AREA, the host will be returned a value of 6 when it reads XMIT QUEUE STATUS. The writes that ended up "off the end" of the memory will not do harm to any data already queued up on the adapter.

The transit frame's destination and source addresses must be explicitly supplied to the adapter for each transmit frame by the host. This information can be provided as part of the immediate data or, if there is no immediate data, as the first fourteen bytes of the first data buffer specified in the descriptor.

Essentially, the host provides every byte of the frame between the start of frame delimiter and the frame check sequence (CRC) through the XMIT AREA register or download DMA operation.

Although in general data must be written to XMIT AREA in the order specified, one exception exists. XMIT BUFFER COUNT/XMIT IMMED LEN may be rewritten after the initial values for these fields have been written, and before any buffer descriptor entries have been written.

The following restrictions apply:

1. When rewriting XMIT BUFFER COUNT/XMIT IMMED LEN, the new value of XMIT IMMED LEN cannot specify a value smaller than the number of immediate data bytes already written to XMIT AREA. It may specify a greater number. If the number is greater, then the remaining immediate data bytes (and any buffer descriptors) shall be written after the new XMIT BUFFER COUNT/XMIT IMMED LEN value is written.

2. When rewriting XMIT BUFFER COUNT/XMIT IMMED LEN, the complete 32 bit quantity must be rewritten, even if one of the values remains the same. The capability to rewrite XMIT IMMED LEN is included to facilitate direct movement of data from one receiving adapter to another transmitting adapter in the host system.

The operating sequence is as follows:

1. The receiving adapter gives an early receive indication to the host.

2. The host, upon determining that the receive frame should be transmitted on the second adapter, sets up a transmit descriptor in XMIT AREA for the second adapter specifying a maximum length, all immediate data frame.

3. The host then sets up a bus master transfer on the receiving adapter that specifies the transmitting adapter as the destination of the transferred data. As data are received, they are bus-mastered by the receiving adapter into the other's XMIT AREA.

4. When the frame reception finishes, the host determines the frame length, and writes that value into the transmitting adapter's XMIT IMMED LEN field.

B. XMIT COMPLETE THRESH

XMIT COMPLETE THRESH provides for an early indication of transmission completion. (Read/write, 4 bytes, 1 32 bit word.)

The XMIT COMPLETE THRESH register is used to specify the number of transmit bytes that remain to be either transmitted or downloaded to the adapter (depending upon the adapter mode) before the adapter will issue a XMIT COMPLETE indication. Only bits **10** through **0** are implemented in this register. Values greater than the maximum frame length will prevent this function from operating properly. This function is disabled by setting the register to zero. The value in XMIT FRAME LENGTH (see below) is used to determine where the end of the transmit frame is.

If this threshold value is set too high, then the host will respond to the indication before the adapter can provide a valid transmit status indication. If XMIT FRAME STATUS returns a ff (hex), then XMIT COMPLETE THRESH should be adjusted to delay the indication slightly. This is accomplished by reducing the value in the XMIT COMPLETE THRESH

register. The function of this register is disabled during the transmission of the first 60 bytes of the frame. This register is cleared to 0 during a reset.

C. XMIT FAILURE

XMIT FAILURE returns the cause of a transmit failure. (Read only, 4 bytes, 1 32 bit word.)

This register returns the cause of the failure of the attempt(s) to transmit a queued frame. A non-zero value indicates that the frame encountered one or more errors during the transmission attempt.

The bits in this register are defined as follows:

bit 0	DMA UNDERRUN
bit 1	LOSS OF CARRIER SENSE
bit 2	MAX COLLISIONS
bit 3	SQE TEST FAILED

This register will contain valid data regardless of the success or failure of the attempt to transmit a frame. If there was no failure, then this register will contain a value of 0 (hex). The contents of this register are valid after the frame has completed transmission (low byte of XMIT FRAME STATUS not equal to ff (hex)) and before XMIT PROT ID is read.

If a data underrun occurs, the adapter will force a CRC error into the frame during transmission to assure that the frame is received as a bad frame and is discarded by the destination device.

D. XMIT FRAME LENGTH

XMIT FRAME LENGTH returns the number of bytes to be transmitted. (Read Only, 4 bytes, 1 32 bit word.)

The XMIT FRAME LENGTH register returns the total number of bytes queued up for transmission by the current transmit frame descriptor identified by CXD pointer. This value is the total of the number of immediate data bytes and of all of the buffer length fields downloaded to the adapter for this frame. The value returned by this register does not reflect the effects of any padding of the frame that may be done by the adapter when the frame is less than 60 bytes in length.

The XMIT FRAME LENGTH register becomes valid immediately after the host writes the last byte to XMIT AREA and remains valid until the first write to XMIT AREA after a read of XMIT QUEUE STATUS.

E. XMIT FRAME STATUS

XMIT FRAME STATUS returns the results of a transmit attempt. (Read only, 4 bytes, 1 32 bit word.)

The least significant 16 bits of this register return the status of the attempt(s) to transmit a queued frame. The most significant 16 bits returns the XMIT REQ HANDLE for the frame. A value of XXXX0000 (hex) (XXXX is the XMIT REQ HANDLE for this particular frame) is returned for a successful transmission while XXXX000a (hex) is returned for a failed transmission. XXXX00fe (hex) is returned d the adapter is in the process of retrying a transmission after a collision. If the transmission is still in progress, XMIT FRAME STATUS will return a XXXX00ff (hex).

If the frame was not transmitted successfully the specific cause of the transmit failure is available in XMIT FAILURE. Reading XMIT PROT ID advances XMIT FRAME STATUS to the status of the next transmitted frame, if any. If the "retry" status value is returned when XMIT FRAME STATUS is read, then reading XMIT FRAME STATUS will also clear the XMIT COMPLETE indication.

F. XMIT PROT ID

XMIT PROT ID returns the protocol ID of the transmit frame. (Read only, 4 bytes, 1 32 bit word.)

As soon as the adapter has completed its attempt(s) to transmit a queued frame and has posted its status, XMIT PROT ID can be read by the host as a method of identifying the frame. The value returned here is the same value that was written into the XMIT PROT ID field during the queuing of the frame via XMIT AREA.

Reading this register clears the XMIT COMPLETE indication except when the "retry" status value is read from XMIT FRAME STATUS. If "retry" was read, then reading XMIT FRAME STATUS will have cleared XMIT COMPLETE.

The XMIT PROT [D value resides in the upper 16 bits of the 32 bit register. The least significant 16 bits of this register will return the MACID value written to the MACID register. A double-word read will return both values simultaneously.

As multiple frames can be queued up for transmission, so can multiple transmission results be queued. Reading both words of XMIT PROT ID advances the completion status in XMIT FRAME STATUS, XMIT FAILURE, and XMIT PROT ID to the status of the next frame which has completed transmission, if any.

G. XMIT QUEUE STATUS

XMIT QUEUE STATUS returns the results of queuing a transmit frame. (Read only, 4 bytes, 1 32 bit word.)

A read of the XMIT QUEUE STATUS register returns the status of the host's attempt to queue up a transmit frame via XMIT AREA.

2 (hex)—Success: If the transmit request was successfully queued, this value is returned when XMIT QUEUE STATUS is read.

6 (hex)—Out of Resources: If the adapter runs out of queue storage RAM, then a status of 6 (hex) is returned.

7 (hex)—Frame Too Long: If the total number of bytes to be transmitted in a single frame exceeds the maximum frame length, this register will return a 7 (hex).

a (hex)—Order Violation: If the data written to XMIT AREA is written out of order, then this error code is returned.

ff (hex)—Adapter Not Ready: If XMIT QUEUE STATUS is read too quickly after the completion of the writes to XMIT AREA, it is possible to read the status value before the queuing process is complete.

Reading this register also advances XMIT AREA so that another transmit may be queued. This register MUST be read after all of the data has been written to XMIT AREA and before the next transmit request is written to XMIT AREA.

If an error code (6 (hex), 7 (hex), a (hex), or ff (hex)) is returned by XMIT QUEUE STATUS, then the frame was not queued and the host will have to attempt to queue it up another time. If the frame was flagged as being too long, it will have to be broken up into multiple frames before another queue attempt can be made.

If the code indicating success (2) is returned, then the host may immediately proceed to attempt to queue up an additional frame. The number of frames that can be queued depends on the amount of RAM on the adapter allocated for this purpose and the amount of immediate data included in each frame.

If the host attempts to queue a frame that is too large and also exceeds the available TRANSMIT DESCRIP-

TOR free space, the error that occurs first will take precedence and will be returned to the host.

H. XMIT START THRESH

START THRESH provides for an early begin of transmission. (Read/write, 4 bytes, 1 32 bit word.)

The XMIT START THRESH register is used to specify the number of transmit bytes that must reside on the adapter before it will start transmission. Only bits 10 through 0 are implemented in this register. Values greater than the maximum frame length will prevent this function from operating properly. The method for disabling this function is to set the register to zero. Bytes are counted starting with the first byte of the destination field of the transmit frame.

The number of bytes considered to be available is the sum of the immediate data written to XMIT AREA by the host and those bytes transferred to the transmit data buffers in the adapter using bus master DMA operations. The transmit request will be posted immediately after XMIT START THRESH transmit frame bytes are made available from the immediate data or when the adapter has bus-mastered XMIT START THRESH—XMIT IMMED LEN bytes onto the adapter.

The number of bytes resident on the adapter must be equal to or greater than the value in XMIT START THRESH for the transmission to commence, unless the total frame size is less than XMIT START THRESH. In that case, the frame will begin transmission when the entire frame has been copied to the adapter. The actual transmission of the frame may be delayed by previous pending transmit frames and by deferrals to network traffic. This register is set to zero during a reset.

I. TRANSMISSION RING MANAGEMENT

FIGS. 10A–10E illustrate the progression of the pointers used in the transmit operation. In FIGS. 10A–10E, a portion of the transmit descriptor ring generally 200 and the transmit data buffers 201-0 and 201-1 are shown. Also, the pointers abbreviated as discussed above are indicated. One might also represent the transmit descriptor ring 200 in a circular fashion, as done below in FIGS. 12A–12F for the receive ring.

In FIG. 10A, the host descriptor logic is writing a first descriptor into the transmit descriptor ring 200. Thus, the CXD pointer points to the base address of the first descriptor, the HW pointer points to the offset from the base address to which the host is expected to write the next double word of the descriptor. The download pointers including CDD, CDB are also pointing to the base address of the first descriptor as no download operations have begun. Likewise, the transmit descriptors CXD and XR point to the same base address. Finally, the tail of the ring XT points to the beginning descriptor. The download data pointer DD points to the top of a first buffer, for instance, buffer zero.

As illustrated in FIG. 10B, the first descriptor has completed writing and has begun the download process, and the host has begun writing a second descriptor. Thus, the host descriptor logic pointer CHD points to the base address of the next descriptor, and the HW pointer points to the expected address of the next byte. The download pointer CDD points to the base address of the first descriptor. The down DMA logic is assumed to be in the process of transferring buffers from host into the transmit data buffer. Thus, the CDB pointer points to a descriptor of a download buffer in the first descriptor, and the DD pointer points to an offset within the transmit data buffer at which data is being downloaded. No transmit operations are yet to begin because the threshold has not been reached. Thus, the transmit pointers and the tail pointer still point to the first descriptor.

In FIG. 10C, the host descriptor logic is working on a third descriptor, the download logic is working on the second descriptor, and the transmit logic is working on the first descriptor. The host descriptor logic pointers CXD and XR are working on the first descriptor. Thus, the CXD pointer points to the base address of the first descriptor, and the XR pointer points to immediate data being read for transmission by the transmit DMA logic.

The transmit read pointer XR will complete reading the immediate data and then move to the transmit data buffer filled by the download DMA logic when processing the first descriptor as illustrated in FIG. 10B.

The transmit tail pointer XT still points to base address of the first descriptor.

The download logic is working on the second descriptor. Thus, the CDD pointer points to the base address of the second descriptor, the CDB pointer points to a buffer descriptor within the second descriptor, the DD pointer points to an offset within the second transmit data buffer to which the download DMA logic is transferring data from the host.

In FIG. 10C, the host is writing a third descriptor into the XMIT AREA register. Thus, the CHD pointer points to the base address of the third descriptor, and the HW pointer points to the off set to which the next byte is expected to be written by the host.

In FIG. 10D, the process has moved on so that the host is writing to a fourth descriptor. Thus, the CHD pointer points to the base address of the fourth descriptor, and the HW pointer points to the expected address of the next write to the XMIT AREA register.

The download logic is working on the third descriptor. Thus, the CDD pointer points to the base address of the third descriptor, the CDB pointer points to a buffer descriptor within the third descriptor, and the download data DD pointer points to a location in the first data buffer at which the download is occurring. This operation assumes that the transmit of the first descriptor has freed up the first data buffer for use by the download logic.

The transmit logic is working on the second descriptor. Thus, the CXD pointer points to the base address of the second descriptor, and the XR pointer points to a location in the transmit data buffer from which data is being read by the transmit logic. Since the status of the first descriptor is yet to be read, the transmit tail XT pointer still points to the base address of the first descriptor.

In FIG. 10E, the process is assumed to have progressed so that the write of the fourth descriptor is completed, but the host has ceased writing new descriptors temporarily. In this case, the CHD pointer and the HW pointer point to the base address of a fifth descriptor waiting for further actions by the host. 1: is assumed that the download process has yet to complete downloading the third descriptor. Thus, the current CDD pointer points to the base address of the third descriptor, and the CDB pointer points to a buffer descriptor within the third descriptor. The DD pointer points to a location in the transmit data buffer to which the download process is downloading data.

In FIG. 10E, it is also assumed that the transmission of the frame identified by the second descriptor is complete, and the transmit logic is waiting for the download

operation on a third descriptor to either complete, or download sufficient data that the transmit logic may begin transmission. Thus, the CXD and the XR pointers point to the base address of the third descriptor.

It is assumed that the host has also read the transmit status of the first descriptor. This has the effect of moving the XT pointer to the base address of the second descriptor as illustrated in FIG. 11E.

This process continues with automatic ring wrap-around of the descriptors handled by the adapter. Also, the underrun condition is monitored and appropriate error signals indicated by the adapter.

V. Receive Process

FIG. 11 is a heuristic diagram of the host interface logic and the network interface logic which is involved in the receive function. The host interface logic includes the upload DMA logic 300 and the view logic 301. The upload DMA logic 300 interfaces with the host through the XFER address block described below. The view logic 301 interfaces with the host through LOOKBUF and related address blocks described below.

The network interface logic includes the receive DMA logic 302. Both the host interface logic and network interface logic interface with the receive ring buffer area 303 in the host independent adapter memory. Also, the upload DMA logic 300 interfaces with the transfer descriptor area 304 within the host independent adapter memory.

This logic is also responsible for maintaining the receive buffer ring. Thus, a plurality of pointers are involved. The upload DMA logic generates a receive tail pointer RT which points to the base address of a frame being uploaded, which is also the tail end of the ring. The view logic generates a current frame pointer CF which points to the base address of LOOKBUF in the adapter memory, or a frame being viewed by the view logic 301.

The receive DMA logic 302 generates a current receive pointer CR, pointing to the base address of a frame being received, a receive write pointer RW pointing to a location where data in the current frame is being written. The receive DMA logic 302 also generates a next receive pointer NR which points to the starting position of a next receive frame during status posting.

An understanding of the receive process can be gained from an understanding of the host address map involved.

A. LOOKBUF

The LOOKBUF register is used by the host to examine and/or transfer received frames. (Read only, 2036 bytes, 509 32 bit words.)

The host can use the LOOKBUF register to examine all of part of a received frame. The reads may be of any width and in any order. LOOKBUF uses the least significant eleven address bits of the host's address bus to select the bytes being accessed. The host can examine a portion of the frame and then use the bus master capabilities of the upload DMA logic 300 to transfer the remaining portion of the frame. Alternately, the host can both examine the frame and transfer the contents to the host memory using memory move instructions.

The received frames are always aligned within LOOKBUF such that the destination address field begins at byte zero of the register. The host cannot assume anything about the data read beyond the end of the current frame. The next receive frame is not guaranteed to be there. The host must use a write to RCV DONE to advance LOOKBUF to the next receive frame buffer.

B. RCV BYTES AVAIL

RCV BYTES AVAIL returns the number of valid bytes in LOOKBUF. (Read only, 4 bytes, 1 32 bit word.)

This register provides a running count of the number of bytes that have been received for the current frame. The maximum value to which this register may count is bounded by the value a register RCV MAX AVAIL (not described further).

When read as bytes, the register will store the value of bits 10 through 8 when bits 7 through 0 are read. This ensures integrity of the read of this dynamic register.

C. RCV DONE

RCV DONE allows LOOKBUF to advance to the next frame. (Write only, 4 bytes, 1 32 bit word.)

A write of an arbitrary value to the least significant byte of this register will cause LOOKBUF to advance to the next receive frame (if any). RCV BYTES AVAIL, RCV FRAME SIZE and RCV FRAME STATUS are all similarly updated.

The frame that was in LOOKBUF at the time that RCV DONE was written to cannot be restored to LOOKBUF. The adapter will preserve the frame both in its internal receive buffer and in LOOKBUF until RCV DONE is eventually written to. Although the data in LOOKBUF is no longer visible once RCV DONE is written to, any data transfers that were initiated by writing to XFER AREA will complete successfully. The received frame's data will be preserved until the data transfer completes.

D. RCV FRAME SIZE

RCV FRAME SIZE returns the size of the current receive frame. (Read only, 4 bytes, 1 32 bit word).

RCV FRAME SIZE returns the size (in bytes) of the current receive frame in the lower 16 bits and the MACID value in the upper 16 bits. The length value is not posted and is therefore invalid until the adapter has completed reception of the frame. While the adapter is in the process of receiving the frame, the register will return XXXX0000 (hex) where XXXX is the MACID value written to the MACID register.

This length value will remain valid until the host writes to RCV DONE. When a RCV DONE is issued, the least significant 16 bits of this register return to 0 (hex). The length value is computed by counting all of the bytes from the first byte of the destination address field to the last byte of the data field, inclusive.

If an indication to the host is generated before the entire frame is received, the frame's size will be posted in this register at the completion of frame reception regardless of the final status of the received frame. The RCV FRAME STATUS register should be examined to determine if the frame was received without errors. This register may contain an incorrect value for frames that are received with errors (such as OVERSIZED FRAME).

E. RCV FRAM STATUS

RCV FRAME STATUS returns the status of the current receive frame. (Read only, 4 bytes, 1 32 bit word.)

RCV FRAME STATUS returns the condition of the current receive frame. This status is not posted, and is therefore invalid, until the adapter has completed reception of the frame.

The bits in this register are defined as follows:

bit 0	DMA OVERRUN
bit 1	ALIGNMENT ERROR
bit 2	BAD CRC
bit 3	RUNT FRAME
bit 4	OVERSIZED FRAME

The contents of this register become valid immediately after the completion of the reception process and remain valid until the host writes to RCV DONE. If the adapter is configured to transfer the frame data to the host as it is being received and the frame is received with an error, the adapter will abort the bus master DMA sequence and return an error code to the host regarding the completion of the transfer. The host can at that time read RCV FRAME STATUS to determine if the frame was indeed defective.

In the event of a receive failure, and if the adapter is either configured to receive bad frames or it is enabled to transfer frames during reception, the host must write to RCV DONE to free up the receive buffer on the adapter.

If the adapter generates an EARLY RCV indication or the LENGTH LEFT THRESH register causes a RCV COMPLETE indication, the adapter will receive the frame and post the receive status regardless of whether or not the frame was received with errors even if the adapter is not configured to receive bad frames. In this event, a write to RCV DONE must occur to acknowledge the receive and to discard the frame.

Reading this register acknowledges RCV COMPLETE.

F. XFER AREA

The XFER AREA register is used to supply the adapter with buffer pointers for the transfer of received frames. (Write only, 1,024 bytes, 256 32 bit words).

The purpose of this register is to tell the adapter where in the host memory system the "current" receive frame should be transferred. This is accomplished by writing one or more sets of pointers to this register.

XFER AREA is a write only register that uses the least significant eleven bits of the host's address bus to determine the function of each parameter. The transfer specification must be written to XFER AREA using the structure of FIG. 8.

The actual data transfer is initiated by a read from XFER QUEUE STATUS. If the adapter was able to accept the transfer request without running out of RAM, then the adapter will begin transferring data to the host's memory. If the adapter runs out of memory in its transfer queue buffer, any additional writes to XFER AREA will be ignored. The host must resubmit any rejected transfer requests.

The "current" receive frame is defined as that receive frame that is presently available to the host via LOOKBUF. Receive frames are handled in the same order that they are received from the network. Once discarded by the host, a received frame may not be examined nor may a transfer be set up for it.

Received frames are disposed of by writing to RCV DONE. Until that write occurs, the current frame remains in LOOKBUF and may be transferred to the host an unlimited number of times. Once disposed of, the frame may not be examined or transferred. The disposal occurs immediately unless a transfer is in progress. If a transfer is in progress, then the disposal of the frame will not occur until the completion of the transfer.

G. XFER COMPLETE THRESH

XFER COMPLETE THRESH provides for an early transfer complete indication based on bytes left to transfer. (Read/write, 4 bytes, 1 32 bit word.)

The XFER COMPLETE THRESH register is used to specify the number of bytes from the end of the frame that must be transferred to the host by the adapter before it will generate a XFER COMPLETE indication to the host. When the number of bytes remaining to be transferred is equal to or less than the value in XFER COMPLETE THRESH, the XFER COMPLETE indication will be set (if not masked). Only bits 10 through 0 are implemented in this register. However, values greater than the maximum frame length will prevent this function from operating properly. The preferred method for disabling this function is to set the register to zero.

If the XFER COMPLETE THRESH condition is met while the frame being transferred is still being received, XFER COMPLETE is suppressed until the frame is received. If XFER COMPLETE is suppressed in this manner, LENGTH LEFT THRESH will override the value in this register and will be used to generate an early XFER COMPLETE indication.

If the frame is fully received when the transfer is initiated, then the transfer length is the lesser of the actual frame length and the number of bytes requested to be transferred via XFER AREA. If the frame is still being received when the transfer is initiated, then its length is assumed to be the value of the length field plus 14. If the value of the length field is greater than 1500, but not equal to 8137 (hex) (i.e., a special frame type identifier), then the frame length is assumed to be 60 bytes. Finally, if the length field contains 8137 (hex), then byte 17 and 18 of the receive frame will be assumed to contain the frame's length value. This assumed length will be changed when the frame is entirely received to the lesser of the actual frame size and the number of bytes requested to be transferred.

This value can be tuned by the host as it determines how often it is responding too early to the transfer complete indication versus how often it is arriving late or on time. If the host is responding too early, XFER STATUS will return a ff (hex) indicating that the status is undefined and that the transfer will be completed shortly. This register is set to 0 during a reset.

H. XFER STATUS

XFER STATUS returns the status of a receive frame transfer. (Read only, 4 bytes, 1 32 bit word.)

Reading this register will return the status of the receive frame transfer attempt. The possible return values are 0 (hex) for success, a (hex) for failure, and ff (hex) for unknown. XFER STATUS is set to ff (hex) while a transfer is in progress. Transfer failures can be caused by excessive wait states encountered on the host bus or by the receive frame being defective. Reading this register will acknowledge (reset) XFER COMPLETE in an INDICATION REASON register (not further described).

I. RECEIVE RING MANAGEMENT

The management of the receive buffer is accomplished with five pointers, as described above with respect to FIG. 11. FIGS. 12A-12F illustrate progression of the pointers for the receive process. Current receive pointer CR points to the base address of the frame being received. Receive write pointer RW points to the location where data is being written to. Next receive pointer NR points to the starting position of the next receive

frame during status posting. Current frame pointer CF points to the base address of LOOKBUF. Receive tail pointer RT points to the base address of the frame being uploaded which is also the tail end of the ring.

After initialization or when no receive frame is being queued in the buffer, all pointers will point to the same location as shown in FIG. 12A.

Before frame reception begins, the contents of the first three locations of a receive buffer in the ring, which are the NEXT RCV PTR, RCV FRAME STATUS, and RCV FRAME SIZE registers, will be cleared. After this is completed, the RW pointer will point to the location where the first byte of a receive frame is to be deposited.

During frame reception, receive data are being assembled into 32 bit words and deposited into the location pointed to by RW pointer. If an address mismatch or frame error occurs and receive indications have not been set during reception, then receive buffer logic will assert RECEIVE REJECT IN to command frame rejection, and RW will be set back to the location of CR. If EARLY RCV becomes set during frame reception, the host can start examining the received data and decide to discard the frame or to command data transfer by upload DMA. If the host decides to discard the frame by writing to RCV DONE, then frame rejection will take place. However, if the host decides to setup upload DMA for data transfer, then it is the responsibility of upload DMA to ensure it does not get ahead of RW pointer.

After the last byte of the receive frame has been received and deposited into the receive buffer, RW pointer will be pointing to the starting location of the next receive frame as shown in FIG. 12B. At this time NR pointer will be set to the location pointed by RW pointer.

During status posting, RW pointer will be set back to the start of the frame just being received as pointed by CR pointer. This is shown in FIG. 12C. At this time, the NEXT RCV POINTER, RCV FRAME STATUS, and RCV FRAME SIZE registers will be updated accordingly.

After the status and size have been posted, CR pointer and RW pointer will be advanced to the location pointed to by NEXT RCV POINTER, and this indicates the end of frame reception. FIG. 12D shows the positions of the pointers at the completion of frame reception.

FIG. 12E shows receive frames being queued in the receive buffer where upload DMA is transferring the "1st frame", LOOKBUF is pointing to the "2nd frame", and incoming receive data is deposited into where RW pointer is pointing. When the host writes to RCV DONE, CF pointer will be updated to point to the beginning of the next frame. In this case, it will be the beginning of the "3rd frame".

When too many frames are being queued up in the buffer and the buffer becomes full, as shown in FIG. 12F, an OUT OF RESOURCES signal will be asserted when CR pointer and RT pointer point to the same location. When this occurs, the frame being received will be rejected, and RW pointer will be set back to where CR pointer is pointing.

VI. Conclusion

The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to practitioners skilled in this art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

What is claimed is:

1. An apparatus for controlling communication between a host system and a network transceiver coupled with a network, wherein the host system includes a host address space, comprising:

a buffer memory outside of the host address space;

host interface means, sharing the host address space with the host, for managing data transfers between the host address space and the buffer memory in operations transparent to the host system; and

network interface means, coupled with the network transceiver, for managing data transfers between the buffer memory and the network transceiver.

2. The apparatus of claim 1, wherein the shared host address space of the host interface means comprises a prespecified block of host addresses of limited size defining a first area for communicating transmit descriptors of transmit frames to the host interface means, and a second area for communicating a view of a received frame through the host interface means to the host system.

3. The apparatus of claim 1, wherein the host interface means includes:

transmit descriptor logic for mapping transmit descriptors identifying data to be transmitted from the host system to the buffer memory; and

download logic, responsive to transmit descriptors in the buffer memory, for retrieving data from memory in the host address space and storing retrieved data in the buffer memory.

4. The apparatus of claim 3, wherein the transmit descriptors include immediate data.

5. The apparatus of claim 1, wherein the host interface means includes:

transmit descriptor logic for mapping transmit descriptors identifying data to be transmitted from the host system to the buffer memory; and wherein the network interface means includes:

transmit logic responsive to transmit descriptors in the buffer memory for retrieving data from the buffer memory and supplying retrieved data to the network transceiver for transmission on the network.

6. The apparatus of claim 5, wherein the transmit descriptors include immediate data.

7. The apparatus of claim 1, wherein the host interface means includes:

transmit descriptor logic for mapping transmit descriptors identifying data to be transmitted from the host system to the buffer memory; and

download logic, responsive to transmit descriptors in the buffer memory, for retrieving data from memory in the host address space and storing retrieved data in the buffer memory; and wherein the network interface means includes:

transmit logic responsive to transmit descriptors in the buffer memory for retrieving data from the buffer memory and supplying retrieved data to the

network transceiver for transmission on the network.

8. The apparatus of claim 7, wherein the transmit descriptors include immediate data.

9. The apparatus of claim 1, wherein the host interface means includes:

transfer descriptor logic for mapping transfer descriptors from the host system to the buffer memory, the transfer descriptors identifying locations in memory in the host address space to which data in the buffer memory is to be stored; and

upload logic, responsive to transfer descriptors in the buffer memory, for transferring data from the buffer memory to memory in the host memory space.

10. The apparatus of claim 9, wherein the network interface means includes:

receive logic for mapping received data from the network transceiver to the buffer memory.

11. The apparatus of claim 1, wherein the host interface means includes:

view logic for presenting data in the buffer memory to the host system prior to transfer to memory in the host address space, in response to host system access to a prespecified portion of the host memory space.

12. The apparatus of claim 1, wherein the host interface means includes:

transmit descriptor logic for mapping transmit descriptors from the host system to the buffer memory, the transmit descriptors including immediate data and data identifying data to be transmitted; and

download logic, responsive to transmit descriptors in the buffer memory, for retrieving data from memory in the host address space and storing retrieved data in the buffer memory;

view logic for presenting data in the buffer memory to the host system prior to transfer to memory in the host address space, in response to host system access to a prespecified portion of the host memory space;

transfer descriptor logic for mapping transfer descriptors from the host system to the buffer memory, the transfer descriptors identifying locations in memory in the host address space to which data in the buffer memory is to be stored; and

upload logic, responsive to the transfer descriptor in the buffer memory, for transferring data from the buffer memory to the host memory space; and

wherein the network interface means includes:

receive logic for mapping received data from the network transceiver to the buffer memory;

transmit logic responsive to transmit descriptors in the buffer memory for retrieving data in the buffer memory and supplying retrieved data to the network transceiver for transmission on the network.

13. An apparatus for controlling communication between a host system and a network transceiver coupled with a network, wherein the host system includes a host address space, comprising:

a buffer memory outside of the host address space, including a transmit buffer and a receive buffer;

host interface means, sharing host address space including a prespecified block of host addresses of limited size defining a first area and a second area, and coupled with the buffer memory, for mapping data addressed to the first area into the transmit buffer, mapping data in the receive buffer into the second area, and uploading data from the receive buffer to the host; and

network interface means, coupled with the network transceiver and the buffer memory, for transferring data from the transmit buffer to the network transceiver and mapping data into the receive buffer from the network transceiver.

14. The apparatus of claim 13, wherein the transmit buffer includes a transmit descriptor ring buffer and a transmit data buffer, and the host interface means includes:

transmit descriptor logic for mapping transmit descriptors from the first area to the transmit descriptor ring buffer and managing breaks in the transmit descriptor ring buffer, the transmit descriptors including data identifying data to be transmitted on the network; and

download logic, responsive to transmit descriptors in the transmit descriptor ring buffer, for retrieving data from memory in the host address space and storing retrieved data in the transmit data buffer.

15. The apparatus of claim 14, wherein the transmit descriptors further include immediate data to be transmitted on the network.

16. The apparatus of claim 14, wherein the network interface means includes:

transmit logic, responsive to transmit descriptors in the transmit ring buffer, for retrieving data from the transmit descriptor ring buffer and transmit data buffer and supplying retrieved data to the network transceiver for transmission on the network.

17. The apparatus of claim 13, wherein the host interface means includes:

transmit descriptor logic for mapping transmit descriptors identifying data to be transmitted from the host system from the first area to the transmit buffer; and

wherein the network interface means includes:

transmit logic responsive to transmit descriptors in the transmit buffer for retrieving data from the transmit buffer and supplying retrieved data to the network transceiver for transmission on the network.

18. The apparatus of claim 17, wherein the transmit descriptors include immediate data to be transmitted on the network.

19. The apparatus of claim 17, wherein the transmit descriptors identify memory in the host address space storing data to be transmitted on the network, and the host interface means further includes:

download logic, responsive to transmit descriptors in the transmit buffer, for retrieving data from memory in the host address space and storing retrieved data in the transmit buffer.

20. The apparatus of claim 13, wherein the receive buffer includes a transfer descriptor buffer and a receive ring buffer, and the host interface means includes:

transfer descriptor logic for mapping transfer descriptors from a third area in the pre-specified block of addresses to the transfer descriptor buffer in the buffer memory, the transfer descriptors identifying locations in memory in the host address space to which data in the receive ring buffer is to be stored; and

upload logic, responsive to transfer descriptors in the transfer descriptor buffer, for transferring data

from the receive ring buffer to memory in the host memory space.

21. The apparatus of claim 20, wherein the network interface means includes:

receive logic for mapping received data from the network transceiver to the receiver ring buffer and managing breaks in the receive ring buffer.

22. The apparatus of claim 13, wherein the host interface means includes:

view logic for presenting data in the receive buffer to the second area for viewing by the host system prior to transfer to memory in the host address space.

23. The apparatus of claim 20, wherein the host interface means includes:

view logic for presenting data in the receive ring buffer to the second area for viewing by the host system prior to transfer to memory in the host address space.

24. An apparatus for controlling communication between a host system and a network transceiver coupled with a network, wherein the host system includes a host address space, comprising:

buffer memory outside of the host address space, including a transmit ring buffer and a receive ring buffer;

host interface means, sharing the host address space with the host, and coupled with the buffer memory, for mapping data from the host system to the transmit ring buffer and to the host system from the receive ring buffer;

network interface means, coupled with the network transceiver and the buffer memory, for mapping data from the transmit ring buffer to the network transceiver and to the receive ring buffer from the network transceiver;

transmit ring pointer means, coupled with the host interface means and the network interface means, for managing accesses to the transmit ring buffer; and

receive ring pointer means, coupled with the host interface means and the network interface means, for managing accesses to the receive ring buffer.

25. The apparatus of claim 24, wherein the transmit ring buffer includes a transmit descriptor ring and a transmit data buffer, and the transmit ring pointer means includes means for generating a plurality of pointers transparently to the host system, the plurality of pointers including:

a host descriptor pointer indicating a location in the transmit descriptor ring at which the host interface means stores a transmit descriptor being written from the host system;

a download descriptor pointer indicating a location in the transmit descriptor ring of a transmit descriptor identifying a frame being downloaded by the host interface means to the transmit data buffer; and

a transmit descriptor pointer indicating a location in the transmit descriptor ring of a transmit descriptor identifying a frame being transmitted on the network.

26. The apparatus of claim 24, wherein the receive ring buffer includes a received frame ring and a transfer descriptor buffer, and the receive pointer means includes means for generating a plurality of pointers transparently to the host system, the plurality of pointers including:

a receive frame pointer indicating a location in the received frame ring to which data being received from the network is being written;

an upload frame pointer indicating a location in the received frame buffer from which data is being uploaded by the host interface means to the host system in response to a transfer descriptor in the transfer descriptor buffer; and

a host view pointer indicating a location in the received frame ring at which data may be read through the host interface means by the host system before uploading.

27. The apparatus of claim 24, wherein the shared host address space of the host interface means comprises a prespecified block of host addresses of limited size defining a first area for communicating transmit descriptors of transmit frames to the host interface means, and a second area for communicating a view of a received frame through the host interface means to the host system.

28. An apparatus for controlling communication between a host system and a network transceiver coupled with a network, wherein the host system includes a host address space, comprising:

buffer memory outside of the host address space, including a transmit descriptor ring buffer, a transmit data buffer, a transfer descriptor buffer and a receive ring buffer;

host interface means, sharing the host address space with the host, and coupled with the buffer memory, for mapping data from the host system to the buffer memory and to the host system from the buffer memory, wherein the host interface means includes

transmit descriptor logic for mapping transmit descriptors from the host system to the transmit descriptor ring buffer, the transmit descriptors including immediate data and data identifying data to be transmitted,

download logic, responsive to transmit descriptors in the transmit descriptor ring buffer, for retrieving data from memory in the host address space and storing retrieved data in the transmit data buffer,

view logic for presenting data in the receive ring buffer to the host system prior to transfer to memory in the host address space, in response to host system access to a prespecified portion of the host memory space,

transfer descriptor logic for mapping transfer descriptors from the host system to the transfer descriptor buffer, the transfer descriptors identifying locations in memory in the host address space to which data in the receive ring buffer is to be stored, and

upload logic, responsive to transfer descriptors in the transfer descriptor buffer, for transferring data from the receive ring buffer to the host system;

network interface means, coupled with the network transceiver and the buffer memory, for mapping data from the transmit ring buffer to the network transceiver and to the receive ring buffer from the network transceiver, wherein the network interface means includes

receive logic for mapping received data from the network transceiver to the receive ring buffer, and

transmit logic responsive to transmit descriptors in the transmit descriptor ring for retrieving data in the transmit descriptor ring buffer and the transmit data buffer and supplying retrieved data to the network transceiver for transmission on the network;

transmit ring pointer means, coupled with the host interface means and the network interface means, for managing accesses to the transmit ring buffer and the transmit data buffer, including means for generating a plurality of pointers, the plurality of pointers including

a host descriptor pointer indicating a location in the transmit descriptor ring at which the transmit descriptor logic stores a transmit descriptor being written from the host system,

a download descriptor pointer indicating a location in the transmit descriptor ring of a transmit descriptor identifying a frame being downloaded by the download logic to the transmit data buffer, and

a transmit descriptor pointer indicating a location in the transmit descriptor ring of a transmit descriptor identifying a frame being transmitted by the transmit logic on the network; and

receive ring pointer means, coupled with the host interface means and the network interface means, for managing accesses to the receive ring buffer including means for generating a plurality of pointers, the plurality of pointers including

a receive frame pointer indicating a location in the receive ring buffer to which data being received through the receive logic from the network is being written,

an upload frame pointer indicating a location in the receive ring buffer from which data is being uploaded by the upload logic to the host system in response to a transfer descriptor in the transfer descriptor buffer, and

a host view pointer indicating a location in the receive ring buffer at which data may be read through the view logic by the host system before uploading.

29. The apparatus of claim 28, wherein the shared host address space of the host interface means comprises a prespecified block of host addresses of limited size defining a first area for communicating transmit descriptors of transmit frames to the host interface means, and a second area for communicating a view of a received frame through the host interface means to the host system.

* * * * *